



**UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR**

**PROYECTO DE FIN DE CARRERA
INGENIERÍA INFORMÁTICA**

**WAEX 2: Validador de Accesibilidad
Web y corrector de errores**

Autor: Sonia Almodóvar Vialas
Tutor: Vicente Luque Centeno

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO	2
ÍNDICE DE FIGURAS	4
ÍNDICE DE TABLAS	5
AGRADECIMIENTOS	7
1. RESUMEN	8
2. CAPÍTULO I: INTRODUCCIÓN	11
2.1. Motivación	11
2.2. Objetivos	16
2.3. Estructura del documento	18
2.3.1. Investigación	18
2.3.2. Análisis	19
2.3.3. Desarrollo.....	19
3. CAPÍTULO II: INVESTIGACIÓN	21
3.1. Estado del arte.....	21
3.2. Selección de los objetivos	28
3.3. Casos de uso.....	30
4. CAPÍTULO III: ANÁLISIS	35
4.1. Alcance del sistema	35
4.2. Identificación de los interesados del sistema	39
4.3. Identificación y definición de los requisitos del sistema	40
4.3.1. Requisitos funcionales	42
4.3.2. Requisitos de interfaz.....	44
4.3.3. Requisitos de rendimiento	47
4.3.4. Requisitos operacionales.....	47
4.3.5. Requisitos de documentación	48
4.3.6. Requisitos de seguridad	48
4.3.7. Requisitos de portabilidad	49
4.4. Identificación de las alternativas.....	50
4.4.1. Entrada al sistema	50
4.4.2. Recorrido de páginas	51
4.4.3. Validación de WAEX y validación de color.....	52
4.4.4. Soporte a cookies	53
4.4.5. Modificación de las páginas.....	54
4.4.6. Salida del sistema.....	54
4.5. Selección de la solución a desarrollar	55
4.5.1. Recorrido de páginas	55
4.5.2. Validación de WAEX y validación de color.....	55
4.5.3. Soporte a cookies	57

4.5.4.	Modificación de las páginas.....	58
5.	CAPÍTULO IV: DESARROLLO	59
5.1.	Escenario.....	59
5.1.1.	Interfaz de entrada al sistema.....	59
5.1.2.	Interfaz principal de validación	62
5.2.	Selección de las herramientas software	64
5.3.	Arquitectura de la solución	67
5.4.	Diseño e implementación de las bibliotecas	71
5.4.1.	Lógica del sistema	71
5.4.2.	Librería ejecutor de cambios.....	79
5.4.3.	Extensión de Mozilla Firefox	81
5.5.	Pruebas de funcionamiento	83
5.5.1.	Recorrido de páginas	84
5.5.2.	Pruebas de validación	86
5.5.3.	Pruebas de soporte a cambios	87
5.5.4.	Pruebas de cambios.....	90
5.5.5.	Prueba de descarga.....	93
6.	CAPÍTULO V: CONCLUSIONES Y LÍNEAS FUTURAS	94
6.1.	Conclusiones	94
6.2.	Líneas futuras.....	95
6.2.1.	Posibles ampliaciones	95
6.2.2.	Alternativas de implementación	97
6.2.3.	WCAG 2.0	104
	ANEXO I Código de una regla en PHP	106
	ANEXO II Código de una regla en C++	110
	ANEXO III INSTALACIÓN Y CONFIGURACIÓN.....	116
1.	Instalación y configuración del sistema.....	116
1.1.1.	Prerrequisitos de instalación	116
1.1.2.	Instalación y configuración.....	116
1.1.3.	Instalación de la aplicación para cada uno de los usuarios.....	117
	ANEXO IV DOCUMENTACIÓN	120
1.	Manual de usuario.....	120
1.1.1.	Introducción a WAEX	120
1.1.2.	Realizar Cambios de color	123
1.1.3.	Realizar Cambios de Validación del Código	125
	GLOSARIO DE TÉRMINOS	126
	Acrónimos	126
	Definiciones.....	128
	REFERENCIAS	130
	Documentos electrónicos.....	130
	Herramientas.....	134

ÍNDICE DE FIGURAS

Figura 1 Inicio de la aplicación actual de WAEX	21
Figura 2 Ejemplo de resultados de WAEX	22
Figura 3 CU-01 Registro de páginas enlazadas a partir de la inicial	30
Figura 4 CU-01 de la funcionalidad completa.....	31
Figura 5 Interfaz de entrada a la aplicación actual WAEX.	51
Figura 6 Salida de la actual aplicación WAEX	56
Figura 7 Salida de la actual aplicación de validación de color Access Keys	57
Figura 8 Esquema de funcionalidad de WAEX2.....	58
Figura 9 Prototipo de entrada a WAEX2 a través de la extensión.	61
Figura 10 Prototipo de entrada al sistema WAEX2 vía http	62
Figura 11 Prototipo de la interfaz de validación de WAEX2.....	62
Figura 12 Ejemplo de interfaz de validación.....	63
Figura 13 Estructura de los diferentes módulos de la aplicación.	67
Figura 14 Esquema de funcionamiento para recopilar los enlaces del dominio	72
Figura 15 Descripción de las variables generales del sistema.....	73
Figura 16 Ejemplo de la funcionalidad para cambiar color.....	74
Figura 17 Ejemplo de salida adaptada para cambiar errores de la regla WCAG1.1	75
Figura 18 Ejemplo de salida adaptada para cambiar errores de la regla WCAG4.3	77
Figura 19 Ejemplo de salida adaptada para cambiar errores de la regla WCAG5.5	77
Figura 20 Ejemplo de salida adaptada para cambiar errores de la regla WCAG13.1	78
Figura 21 Opción para solucionar problemas de uso del elemento obsoleto FONT	79
Figura 22 Diagrama de clases de la librería de cambios	80
Figura 23 Interfaz de entrada a partir de la extensión de Mozilla Firefox	81
Figura 24 Estructura de mejora de la aplicación	99
Figura 25 Instalación de la extensión WAEX2 para Mozilla Firefox	117
Figura 26 Instalación de la extensión WAEX2 para Mozilla Firefox (completada)	118
Figura 27 Instalación de la extensión WAEX2 para Mozilla Firefox (1)	118
Figura 28 Instalación de la extensión WAEX2 para Mozilla Firefox (2)	119
Figura 29 WAEX2 en la barra de herramientas de Mozilla Firefox	120
Figura 30 Interfaz de entrada a WAEX2 a través de la extensión.....	121
Figura 31 Ejemplo de salidas de WAEX2.....	122
Figura 32 Selección de una página para ser validada.....	123
Figura 33 Leyenda de la validación de colores.	124
Figura 34 Ejemplo para modificar colores de una página	124

ÍNDICE DE TABLAS

Tabla 1 Reglas AAA+	10
Tabla 2 Objetivos de las tareas de investigación	18
Tabla 3 Objetivos de las tareas del análisis	19
Tabla 4 Objetivos de las tareas de desarrollo	20
Tabla 5 Comparativas de los principales validadores Web	24
Tabla 6 Fórmula para el cálculo de diferencia de brillo	27
Tabla 7 Fórmula para el cálculo de la diferencia de contraste	27
Tabla 8 Definición final y detallada de los objetivos	29
Tabla 9 CU-01 Registro de páginas enlazadas a partir de la inicial	31
Tabla 10 CU-02: Validar página con WAEX.....	32
Tabla 11 CU-03: Validar color utilizado en una página.....	32
Tabla 12 CU-04: Facilitar cambios en los errores de validación.	33
Tabla 13 CU-05: Descarga de la solución con los cambios realizados.	33
Tabla 14 Descripción de la nomenclatura de requisitos	40
Tabla 15 Descripción del estilo de tabla para la definición de los requisitos.....	41
Tabla 16 RSF-01 Listado de todas las páginas enlazadas	42
Tabla 17 RSF-02 Validación de una página por WAEX.....	42
Tabla 18 RSF-03 Validación del color de una página.....	43
Tabla 19 RSF-04 Permitir cambios post-validación en WAEX.....	43
Tabla 20 RSF-05 Descarga de las páginas validadas y corregidas	43
Tabla 21 RSF-06 Almacenamiento y descarga de la solución	44
Tabla 22 RSI-01 Dos tipos de interfaces para los dos resultados de validación.	44
Tabla 23 RSI-02 La interfaz permite seleccionar qué página validar.	44
Tabla 24 RSI-03 Interfaz de entrada con los mismos datos de entrada que WAEX.....	45
Tabla 25 RSI-04 Interfaces de entrada al sistema	45
Tabla 26 RSI-05 Interfaz de validación con todas las funcionalidades.....	45
Tabla 27 RSI-06 Espacio de página restringido por el estándar W3C	46
Tabla 28 RSI-07 Estructura de la interfaz de validación.....	46
Tabla 29 RSR-01 Reducir al máximo el tiempo de respuesta de WAEX.	47
Tabla 30 RSO-01 Modos de ejecución de la aplicación.....	47
Tabla 31 RSO-02 El sistema funciona en un servidor con sistema operativo Linux.	48
Tabla 32 RSD-01 Manual de usuario	48
Tabla 33 RSP-01 Acceso al sistema desde cualquier navegador.	49
Tabla 34 Descripción de los campos de entrada a WAEX.....	60
Tabla 35 DTD del fichero de cambios	70
Tabla 36 Prueba P-01	84
Tabla 37 Prueba P-02	84
Tabla 38 Prueba P-03	84
Tabla 39 Prueba P-04	85
Tabla 40 Prueba P-05	86
Tabla 41 Prueba P-06	86
Tabla 42 Prueba P-07	87
Tabla 43 Prueba P-08	87
Tabla 44 Prueba P-09	87

Tabla 45 Prueba P-10	88
Tabla 46 Prueba P-11	88
Tabla 47 Prueba P-12	88
Tabla 48 Prueba P-13	89
Tabla 49 Prueba P-14	90
Tabla 50 Prueba P-15	90
Tabla 51 Prueba P-16	91
Tabla 52 Prueba P-17	91
Tabla 53 Prueba P-18	91
Tabla 54 Prueba P-19	92
Tabla 55 Prueba P-20	93
Tabla 56 Análisis CLIENTE-SERVIDOR	98
Tabla 57 Estimación de horas para interfaces	102
Tabla 58 Estimación de horas para C++	103

AGRADECIMIENTOS

Después de cinco cursos y cinco años puedo afirmar que esta ha sido la mejor etapa de mi vida. En este último escalón, presento por fin el resumen de lo que he aprendido, y dejo con cierta nostalgia mi vida universitaria. Creo que puedo considerarme afortunada en cómo me han ido las cosas en estos años, que sin demasiadas desilusiones he podido sacar año por año las asignaturas que me proponía. Este resultado no hubiese sido posible sin todas las personas que he tenido a mi lado durante este tiempo y me han ayudado a conseguirlo. Es por esta razón que me gustaría agradecerse a todos ellos. Necesitaría bastante más de una página para nombrarlos a todos, y como esto no va a ser posible, me gustaría reconocer el valor especial de algunos de ellos.

En primer lugar, a mis padres, que seguramente con mucho esfuerzo, han aguantado cada una de las épocas de exámenes y me han enseñado a luchar por todo lo que deseo.

A todo el departamento GRI de la Universidad, quiénes consiguieron despertar mi interés por el que ahora es objetivo de este proyecto, la Accesibilidad Web.

A Sergio, quién tuve la suerte de conocer en mis últimos años de carrera y que espero que día tras día pueda seguir devolviéndole todo lo que me ha ayudado.

Y volviendo la vista atrás, no puedo olvidar a todos esos compañeros que desde el primer día han sido parte de esto, y a quienes debo muchos buenos momentos. A todos esos CEBOLLITAS, con quiénes espero seguir pasando muy buenos momentos.

A mis mejores compañeras de prácticas, Nieves y Rebeca. En especial a Nieves, porque debería partir mi título en dos y darle una mitad a ella.

Gracias a Javi por ayudarme leyendo este proyecto.

Y por último, quiero dar mis más sinceros agradecimientos a Vicente Luque, por haberme dirigido este proyecto de fin de carrera.

1. RESUMEN

WAEX [1] son las siglas de Web Accessibility Evaluator in a single XSLT file [2], y cuyo significado en español es: Evaluador de Accesibilidad Web en ficheros XSLT. Esta aplicación está disponible a través de Internet. Su función es aplicar las reglas necesarias para decidir si una página web cumple con las reglas de accesibilidad impuestas por la W3C [3] y publicadas en la primera versión WCAG 1.0 [4]. La W3C [3] (World Wide Web Consortium) es una entidad dedicada a publicar y definir un estándar de accesibilidad. Este estándar está compuesto de un conjunto de reglas cuyo objetivo es ayudar a identificar y solucionar posibles errores de accesibilidad. El estándar tiene todas las reglas clasificadas por su temática. A su vez, cada una de ellas tiene asociado un nivel de accesibilidad y una categoría. Actualmente, y desde diciembre de 2008, la W3C [3] recomienda seguir el estándar WCAG 2.0 [5] del que se comentarán las principales diferencias en los siguientes puntos. En las referencias puede consultarse los enlaces que llevan a las páginas oficiales donde están publicados estos estándares, WCAG 1.0 y WCAG 2.0, en [4] y [5].

Los niveles de accesibilidad deciden cómo de accesible es una página, dependiendo de si cumple con las reglas de las que se compone cada uno de éstos. Por orden ascendente, es decir, cuanto menor es el nivel las reglas son más prioritarias. Una regla es más prioritaria cuando su incumplimiento puede afectar en mayor medida a la hora de acceder a la información de una página. Los niveles definidos son:

- A
- AA
- AAA

La validación se puede realizar en torno a uno de estos niveles. WAEX [1] tiene soporte para validar los niveles anteriores y uno adicional, el AAA+. Este nivel engloba todos los anteriores y añade nuevas reglas que no están definidas como WCAG pero que fueron consideradas como necesarias cuando se pensó en la creación de WAEX [1].

Estas reglas están definidas en el fichero `wcag.xml` [2], que utiliza WAEX [1] para obtener los errores de validación, y se han recogido en la Tabla 1:

Texto mostrado por WAEX	Descripción del error
Maps forbidden	No está permitido el uso de elementos <i>map</i> .
Frames forbidden	No está permitido el uso de elementos <i>frame</i> , <i>frameset</i> o <i>iframe</i> .
Applets forbidden (use object instead)	No está permitido el uso de Applets.
Link's target	La regla 3.19 es ampliada con los elementos <i>frame</i> .
XHTML Basic	No se recomienda el uso de tablas anidadas.
Applets with no code	No se permite el uso de elementos <i>applet</i> que no contengan código.
Badly placed legend elements	Los elementos <i>legend</i> deben estar definidos como primer elemento del nodo que lo contiene.
Badly placed param elements	Los elementos <i>param</i> deben preceder a otros elementos.
Form fields with no name attribute	Todos los elementos <i>textarea</i> , <i>select</i> e <i>input</i> cuyo atributo <i>type</i> no tenga como valor <i>image</i> , <i>reset</i> , <i>submit</i> o <i>button</i> , deben tener en su definición el atributo <i>name</i> .
Non image input with alt attribute	Los elementos <i>input</i> no deben tener atributo <i>alt</i> , a menos que contengan el atributo <i>type</i> y su valor sea <i>image</i> .
Forbidden elements in pre	No se permite el uso de elementos <i>img</i> , <i>object</i> , <i>big</i> , <i>small</i> , <i>sub</i> o <i>sup</i> como hijos de un nodo <i>pre</i> .
Nested focusable elements	No se permite el uso de elementos enfocables anidados.
Nested form	No se permite el uso de elementos <i>form</i> anidados.
Multiplied checked radio buttons	Los grupos de elementos <i>radio buttons</i> solamente deben contener a uno de ellos como seleccionado.
Multiple selected options	Solo debe existir un elemento seleccionado dentro de una definición del elemento <i>select</i> a menos que éste sea <i>multiple</i>
Names different that ids	Los elementos <i>a</i> , <i>applet</i> , <i>object</i> , <i>form</i> , <i>frame</i> , <i>iframe</i> , <i>img</i> y <i>map</i> deben tener los atributos <i>name</i> e <i>id</i> con valores distintos.

Labels with too many form fields inside	Los elementos <i>label</i> solo pueden contener un elemento de tipo <i>textarea</i> , <i>select</i> o <i>input</i> .
Labels not immediately preceding their described form field	Los elementos <i>label</i> deben ir justo delante del campo del formulario que describen.
Different definitions for the same text	Todos los elementos <i>acronym</i> cuyo contenido sea el mismo, deben tener el mismo atributo <i>title</i> .
Deprecated style attribute avoiding consistent style	Los estilos deben ser definidos a través de hojas de estilo externas al documento.

Tabla 1 Reglas AAA+

Las categorías se asocian en función del modo de detección de una regla, o lo que es lo mismo, si una regla es posible aplicarla de forma automática o no. Una regla clasificada como automática quiere decir que el propio validador es capaz de detectar si se cumple o no. Si una regla no puede ser claramente identificada por el validador, debido a que se basa en la semántica del contenido o tiene un carácter opcional, se trata de una regla semiautomática o no automática. En estas últimas será necesario de la razón humana para decidir si es accesible en dicho punto o no.

Para el estudio previo del proyecto, se han analizado los diferentes validadores que hay disponibles en el Web y han sido comparados con WAEX [1] con el fin de desarrollar una herramienta de validación más potente.

WAEX [1] fue desarrollada por Vicente Luque Centeno. Se puede acceder a la aplicación en [1].

Tras un exhaustivo estudio, se pide implementar un programa siguiendo la filosofía de ejecución de WAEX [1] que mejore la actual aplicación.

2. CAPÍTULO I: INTRODUCCIÓN

2.1. Motivación

La importancia de realizar páginas Web accesibles ha sido motivada debido a que una parte significativa de la población tiene problemas de accesibilidad y usabilidad en la Web. La accesibilidad es el grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas, cognitivas o físicas. La usabilidad es la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. En este proyecto, se referirán siempre estas capacidades a la accesibilidad y usabilidad en la Web.

Dichos problemas pueden proceder de discapacidades visuales, auditivas o cognitivas, o simplemente puede resultar inaccesible a toda la población debido a un mal formato de la presentación de la información.

Además, es necesario contar con que los accesos a la información Web puede darse a través de distintos dispositivos como pueden ser las PDA's y móviles. Estos dispositivos tienen limitaciones como por ejemplo, el tamaño del navegador, la interactividad entre usuario y página Web, o el alto procesamiento de los datos. Las personas que acceden a los datos a través de estos dispositivos forman otro conjunto de usuarios que deben ser tenidos en cuenta a la hora de diseñar una página.

Cada vez de forma menos habitual, pero no por ello menos importante, existen usuarios con una conexión a internet lenta, y que les dificulta el acceso a una página que tienen un alto tiempo de carga. Esto puede deberse a una gran cantidad de elementos multimedia. Esto convierte a los elementos multimedia en un punto de estudio de cara a la accesibilidad Web.

Actualmente, en algunos países, existen regulaciones para crear páginas Web. Por ejemplo, los proyectos en los que uno de los clientes es el Gobierno de Estados Unidos, tiene un requisito legal que afecta al diseño de la accesibilidad. La sección 508 de la “Rehabilitation Act” obliga a que toda la información electrónica necesaria para el gobierno de los Estados Unidos sea accesible.

La accesibilidad Web debería ser un requisito en todos los proyectos de desarrollo de sitios Web. Estas reglas no son tan diferentes a las que se aplican en la vida cotidiana. Por ejemplo, los subtítulos en los programas de televisión para que aquellas personas con algún tipo de discapacidad auditiva puedan “escuchar” esa información. Además, estas reglas no sólo ayudan a personas con discapacidad, estos subtítulos podrían ser útiles para lugares donde hay demasiado ruido y no se puede escuchar la televisión. De la misma forma que estas reglas cotidianas pueden ayudar a toda la población, puede darse también en el ámbito de la Web. Por ejemplo, incorporar una descripción a una imagen permite a personas con problemas visuales conocer la información que proporciona una imagen. Esto también afectaría en los casos en los que la imagen en cuestión no es accesible o el tiempo de carga sea muy grande. De este modo se puede leer la descripción de la imagen, en lugar de verla.

A continuación, se muestra una lista de todas las consideraciones que se deberían tener en cuenta a la hora del diseño de un sitio Web:

Personas con dificultades físicas.

Existe un amplio conjunto de discapacidades de este tipo, pero este punto se centra principalmente en los esfuerzos en la entrada de datos a través del teclado y del ratón, porque estas son las principales formas en una entrada para el futuro próximo. Una persona con este tipo de discapacidad podría no ser capaz de usar un teclado de forma eficiente o incluso no poder usarlo en absoluto. Lo mismo ocurre para el uso del ratón. Para solucionar esto, es necesario que cada Web pueda usarse sin ratón, minimizar en todo lo posible el uso de entradas de datos y garantizar botones grandes o posibilidad de hacerlos más grandes para facilitar la tarea.

Personas con discapacidades auditivas.

Esto afecta a todos los sitios que incorporan audio y video. En muchas ocasiones, incorporar audio es únicamente con el fin de ambientar el sitio, es decir, es sólo decorativo. El problema viene cuando se proporciona información a través de sonido o en vídeos. En estos casos, es necesario incorporar elementos textuales para que aquellas personas con discapacidades auditivas tengan acceso a la información que proporcionan estos elementos.

Personas con discapacidades visuales.

Esta parte de la población incluye personas ciegas, con visión limitada o daltónicas. Los objetivos en estos casos es facilitar la lectura de texto y enlaces. Para ello es necesario tener en cuenta que debe haber suficiente contraste entre el color del texto y el del fondo, al igual que ocurre con los enlaces. Hay que dar la posibilidad de usar fuentes grandes, para lo cual lo mejor es utilizar tamaños de letra relativos para que ésta cambie según la configuración de letra que está seleccionada en el navegador Web utilizado por el usuario. Además, debe evitar un texto escrito exclusivamente con letras mayúsculas porque éstas complican su lectura. Las letras mayúsculas sólo deberían ser utilizadas para atraer la atención del lector en algún punto importante y siempre que lo requieran las reglas de ortografía. Las imágenes con animaciones permanentes no deben usarse porque distraen la atención del usuario y dificultan la lectura. Del mismo modo, se debe evitar utilizar texto que aparezca por la parte izquierda de la pantalla y se dirija a la parte derecha. Hay que tener mucho cuidado con la información que hace referencia a un color utilizado. Para estos casos es necesaria una explicación textual para aquellas personas que no pueden distinguir colores o quienes accedan mediante dispositivos con limitación de colores. Por último, hay que tener en cuenta la combinación de colores utilizada para los estados de los enlaces, porque podría ocurrir que las personas no distingan entre cuando un enlace ha sido visitado y cuando no. Además, mantener los colores habituales en el estado de los enlaces hace más fácil su uso puesto que no precisa de un aprendizaje previo.

En el caso de las personas completamente ciegas disponen de un software especial llamado “screen reader”, el cual toma todo el texto de la página y lo transforma en información sonora. Por este motivo es necesario separar el contenido del diseño, para que este programa sea capaz de leer todo el contenido como si únicamente fuese texto. Para estos sistemas, también es importante dar solución a otros elementos no textuales, como por ejemplo multimedia, tablas, frames..., que necesitarán de un texto alternativo para ser traducidos.

Discapacidades cognitivas

Esta discapacidad puede hacer complicado las tareas de escribir, leer o navegar a las personas que la padecen. Para facilitar la navegación se debe colocar la información ordenada, los enlaces deben colocarse siguiendo una estructura y su descripción debe ser clara y definir de forma breve a qué información lleva. En este caso también se deben evitar los elementos que puedan distraer la atención de la información importante, y minimizar la cantidad de información proporcionada por cada página o sección para hacer más sencilla su lectura. También es aplicable a los textos alternativos en imágenes y tablas, que no deben ser demasiado largos, pero sí proporcionar una descripción exacta.

Uso de la Web a través de dispositivos móviles.

El problema de estos dispositivos es que son pobres en cuanto a las capacidades de tamaño, potencia de procesamiento, duración de la batería y entradas de datos, respecto del navegador de un ordenador. Todas estas características deben tenerse en cuenta para el diseño de cada sitio. Por ejemplo, se deben expresar los tamaños con medidas relativas para que puedan ser adaptados a estos dispositivos y minimizar la carga de elementos. Una de las técnicas que pueden ser utilizadas es proporcionar distintos estilos a cada web, que se utilizarán dependiendo del dispositivo desde el cuál se accede. Se debe proporcionar además, alternativas para los scripts, applets y plug-ins que pueden no ser soportados por el dispositivo.

Todos estos problemas están recogidos en forma de reglas que garantizan la accesibilidad Web para todas las personas. Estas reglas se encuentran publicadas en el World Wide Web Consortium (W3C [3]) e intentan regularizar el diseño del Web.

Para garantizar que una página es accesible, debe pasar una validación que compruebe el cumplimiento de cada una de las reglas. Generalmente, estos validadores se basan en la aplicación de reglas de un fichero XSLT. Esta validación debe adaptarse a las reglas publicadas por la W3C [3].

Existen multitud de validadores capaces de garantizar la accesibilidad. Pueden encontrarse como aplicaciones instalables o disponibles online. El principal objetivo del proyecto será mejorar el validador WAEX [1].

2.2. Objetivos

El objetivo principal es la mejora de la funcionalidad de WAEX [1]. Se propone un proyecto consistente en la búsqueda e implementación de las principales características que pueden ser mejoradas o incorporadas a la actual aplicación. Estas características podrán ser nuevas aportaciones o funcionalidades que incorporan otros validadores.

A pesar de la libertad en el desarrollo, se pide que WAEX [1] mantenga sus características de ser una herramienta disponible en Web y pueda seguirse utilizando como vía alternativa al nuevo proyecto, es decir, la herramienta WAEX [1] no se debe modificar internamente, sino que se incorporarán las mejoras en una capa externa a WAEX [1] y que sea transparente al usuario. De este modo, si se modifica el XSLT a partir del cual funciona WAEX [1], esto no influirá en el funcionamiento de la nueva aplicación.

El objetivo del proyecto a nivel de usuario, es facilitar el uso de este validador y el trabajo de los desarrolladores Web, y colaborar de este modo en el desarrollo de sitios accesibles.

Tras la investigación desarrollada, que será detallada en los siguientes apartados, se han destacado aquellos objetivos que han sido seleccionados como las metas más importantes a conseguir. Estos objetivos son los siguientes:

- Permitir que la herramienta sea capaz de validar todas las páginas del mismo dominio a las cuales se acceden mediante hipervínculos desde la URL enviada al validador.
- Facilitar los cambios que deben hacer los usuarios como medio de corrección de los errores de validación, proporcionando entradas de datos u opciones de cambio. También se debe dar la oportunidad de tener una visualización de éstos siempre que sea posible.
- Tener soporte a páginas cuya entrada sea mediante uso de cookies.

- Mejorar la hoja de estilos según los errores de validación.
- Incorporar todos los cambios mediante una capa externa a WAEX [1] que funcione realizando llamadas al sistema ya desarrollado.
- Para estas mejoras, será imprescindible un estudio de los distintos validadores que se encuentran en el mercado. De éstos, los que realmente interesan para el estudio son los que están disponibles online, es decir, no es necesario instalar nuevas aplicaciones antes de validar el sistema.
- La búsqueda de estos validador se ha realizado a partir de una búsqueda desde la W3C [3] que se indica en la bibliografía del documento.

2.3. Estructura del documento

En este documento se podrá consultar cualquiera de las tareas seguidas en el desarrollo del proyecto. Debido a que es un proyecto de investigación y ampliación sobre el entorno actual, se fijan los objetivos tras el análisis del Estado del Arte, que coincidirán con los citados en el punto anterior. Se consideran los requisitos del sistema en función de los objetivos y metas propuestos, todos ellos con aprobación de Vicente Luque Centeno, tutor del proyecto.

Se ha decidido dividir la estructura del documento en cuatro bloques fundamentales. Se pretende seguir un orden cronológico a la hora de describir todos los detalles del proyecto, desde la propuesta final y la investigación hasta el producto final.

2.3.1. Investigación

En el primer bloque se explicará todo el proceso de investigación acerca de los validadores web, así como las pautas que se han seguido para seleccionar los objetivos. Hay que destacar, que la investigación ha sido mayor de la que se detalla en este documento, pero únicamente se ha mantenido la información relevante para la solución final. Este bloque se corresponde con el CAPÍTULO II: INVESTIGACIÓN de este documento. A continuación, en la Tabla 2, se pueden ver las tareas de las que forma parte, las cuáles serán descritas en los subapartados del mismo capítulo.

Tarea	Objetivo
Estado del Arte	Esta tarea tiene como fin estudiar la situación actual. Analizar los sistemas existentes en el mercado, con funcionalidad semejante a la del producto que se quiere desarrollar.
Selección de los objetivos	Es la definición y descripción de todos los objetivos y funcionalidades nuevas que se pretenden implementar para la nueva aplicación.
Casos de uso	Es una técnica para capturar y representar gráficamente los requisitos principales del sistema.

Tabla 2 Objetivos de las tareas de investigación

2.3.2. Análisis

Una vez conocidas las metas, se analizan en detalle todos sus objetivos y características, con el fin de obtener los requisitos del sistema y la identificación de las posibles soluciones según estos, todo ello englobado en el **CAPÍTULO III: ANÁLISIS**. Las tareas de las que constará la parte de análisis y elección de las posibles soluciones son las definidas en la Tabla 3:

Tarea	Objetivo
Identificación del alcance del sistema	Indicar las capacidades del sistema a desarrollar, es decir, qué es lo que va a resolver.
Identificación de los interesados en el sistema	Identificar todos los interesados (personas u organizaciones involucradas activamente en el proyecto) para poder determinar su participación tanto en el estudio de la situación actual como en la toma de requisitos de usuario, así como en el resto del proceso de desarrollo.
Identificación y definición de requisitos	Obtener una definición de lo que el usuario espera que realice el sistema informático mediante reuniones con los interesados. Se recogerá un catálogo de requisitos.
Identificación de las alternativas	Estudiar las diferentes alternativas que hay para configurar la solución de forma que se responda satisfactoriamente a los requisitos planteados.
Valoración de las alternativas	Analizar las alternativas desde el punto de vista económico, tecnológico, organizativo y de los riesgos.
Selección de la solución	Seleccionar la solución final o determinar la inviabilidad del sistema por motivos económicos, de funcionalidad, por no poder cubrir o dar respuesta a los requisitos en un plazo razonable de tiempo, o por otro motivo que se considere de entidad suficiente como para no continuar adelante.

Tabla 3 Objetivos de las tareas del análisis

2.3.3. Desarrollo

El tercer bloque continuará con la descripción del proceso de desarrollo del sistema a un nivel de implementación, manteniendo el diseño de la arquitectura de la solución elegida en el bloque anterior. También se decidirán todas las características de la

interfaz gráfica. Por último, se destacarán algunas de las características más importantes en la implementación. De forma más específica, las tareas en las que se divide este Capítulo IV son las que se definen en la Tabla 4:

Tarea	Objetivo
Escenario	Descripción del entorno y sus funcionalidades.
Selección de las herramientas software	Listado de todas las herramientas software, así como lenguajes de programación, que formarán parte de la solución final.
Arquitectura de la solución	Diseño de la estructura organizada en módulos independientes previa a la implementación.
Prototipos	Descripción gráfica de todas las interfaces de usuario del sistema.
Diseño de bibliotecas	Diseño de los distintos módulos que forman parte del sistema, siguiendo la arquitectura de la solución anterior.

Tabla 4 Objetivos de las tareas de desarrollo

El quinto capítulo del documento está destinado a las conclusiones y las líneas futuras cuyo fin es dar a conocer qué se podría mejorar en el proyecto. En este último capítulo se tratarán los temas relacionados al momento posterior al desarrollo del sistema, es decir, una vez terminado el proceso de implementación.

Por último, como ayuda para la lectura de este documento, se muestra el glosario de términos, referencias y los anexos con información que complementa a este documento para que finalmente el lector pueda tener una idea completa del proyecto. Existen cuatro anexos, dos de ellos en los cuáles se especifica el ejemplo del código de una regla en PHP y otro con la implementación en C++, información adicional al CAPÍTULO V: CONCLUSIONES Y LÍNEAS FUTURAS.

3. CAPÍTULO II: INVESTIGACIÓN

3.1. Estado del arte

Ya que la finalidad del proyecto es obtener una mejora de un sistema ya existente, el primer paso será repasar todas las características y funcionalidades que posee WAEX [1].

Como ya se ha comentado anteriormente, WAEX [1] son las siglas de *Web Accessibility Evaluator in a single XSLT file*, o por su significado en castellano, es un validador de accesibilidad Web que decide si una página facilita el acceso a la información por cualquier tipo de persona. El resultado de la validación será una nueva página HTML [7] con un resumen de los errores.

WAEX [1] ha sido implementado por Vicente Luque Centeno. La interfaz de entrada a la aplicación se muestra en la Figura 1.

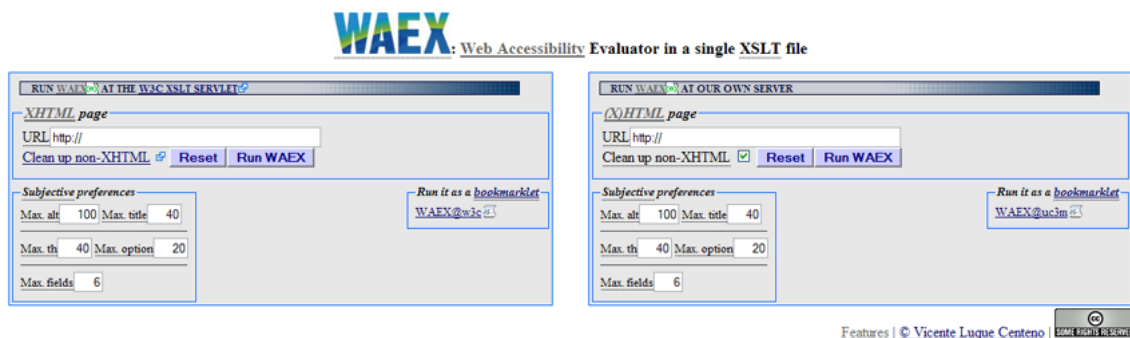


Figura 1 Inicio de la aplicación actual de WAEX

La entrada a WAEX [1], según se muestra en la Figura 1, tiene dos posibilidades que coinciden con cada uno de los cuadros de la imagen. La diferencia entre ambos reside en el servidor en el que será ejecutado el fichero XSLT. En el cuadro de la izquierda, el XSLT que implementa las reglas será ejecutado en el servidor de la W3C [3], mientras que en el de la derecha será ejecutado en el mismo servidor donde se encuentra este fichero, en <http://www.it.uc3m.es>.

Entre las opciones de la parte inferior de ambos cuadros se encuentran las opciones Max. alt., Max. title., Max. th., Max. option. y Max. fields utilizadas para limitar el tamaño de los atributos o elementos que se describen en su nombre.

Una vez se pulsa sobre el botón *Run*, WAEX [1] comienza la validación. La página es sometida internamente a un fichero xslt. El fichero con las reglas que utiliza WAEX [1] se puede consultar a través del siguiente enlace que se indica en la referencia [2]. Las reglas que valida son las publicadas por la W3C [3] para garantizar la mayor accesibilidad definida en dicho estándar.

La salida que se obtiene al aplicar el fichero wcag.xsl [2] con las reglas implementadas, es una nueva página con un estilo similar al siguiente ejemplo (Figura 2), dependiendo de los errores que se hayan encontrado.

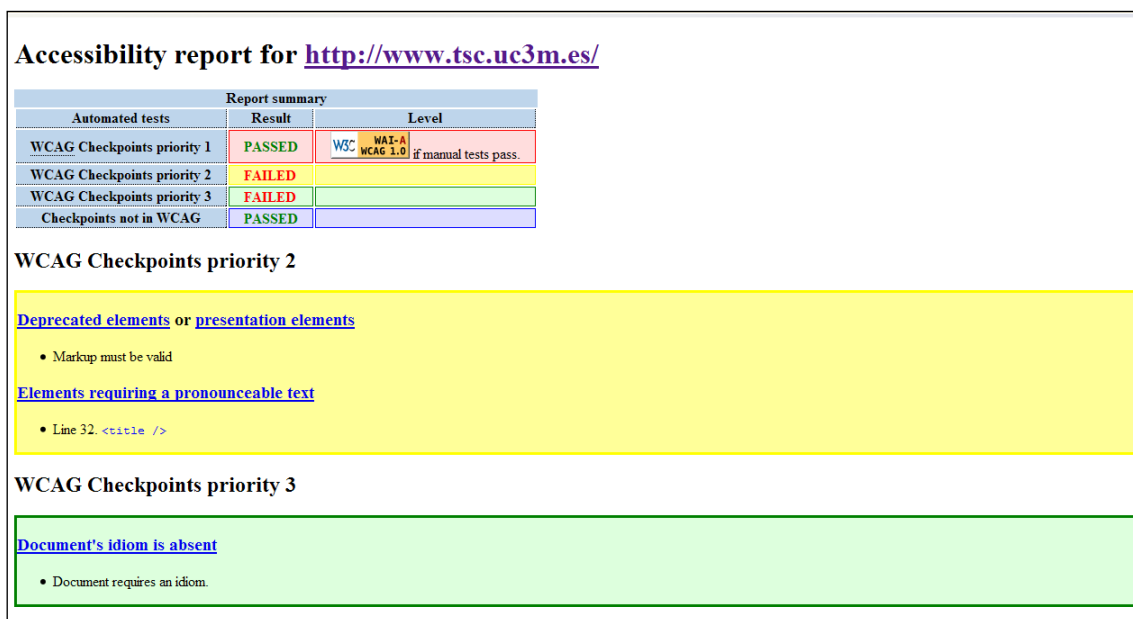


Figura 2 Ejemplo de resultados de WAEX

En la parte superior aparece una tabla donde se indica qué niveles supera (prioridad A, AA o AAA) o cuales han fallado. Los errores encontrados según las reglas de validación, son clasificados y mostrados, según la prioridad de la regla, dentro de un mismo recuadro con un color característico. Cada uno de estos niveles se caracteriza con un recuadro con un color específico. Así, las reglas fallidas de prioridad 1 se

mostrarán dentro de un cuadro rojo, las de prioridad 2 en un cuadro amarillo y las de prioridad 3 en un cuadro verde.

Una de las principales características y más importantes es que está al alcance de cualquier desarrollador Web, ya que no necesita de instalación y puede utilizarse desde la misma página donde está publicado, en la referencia [1].

Una vez se ha realizado un resumen de la funcionalidad de WAEX [1], se procede al estudio de esta tarea. El objetivo es obtener la información acerca de los diferentes validadores que hay disponibles actualmente. En concreto, aquellos que están presentes en Web y de forma gratuita. Se recopila información acerca de los servicios y capacidades que cada uno de ellos aporta y se compararan estos validadores entre ellos y frente a las nuevas necesidades del validador que se pretende mejorar, WAEX [1]. Se recopila información acerca de los servicios y capacidades que cada uno de los seleccionados aporta, para posteriormente, realizar una comparativa entre estos validadores y recoger las nuevas necesidades de WAEX [1].

La documentación que se comenta a continuación ha sido recopilada de la siguiente página Web, donde se encuentra un amplio listado de los validadores y sus funcionalidades, consultado en el recurso electrónico de la referencia [6], en la que se puede ver un listado de todos los validadores de accesibilidad Web que han sido publicados.

Las características que se pueden recoger directamente acerca de cada uno de ellos se resumen en la Tabla 5. Se han diferenciado en la tabla unas filas respecto de otras por el color de fondo, cuyo significado es el siguiente:

- **Naranja** para validadores que comprueban si los colores utilizados son apropiados, en cuanto a contraste y brillo.
- **Verde** para validadores que únicamente tratan las características comprobables acerca de las imágenes.
- **Sin color** para el resto.

	Descripción	Language	Guidelines	Automatic checking	Formats	Reports	Observaciones
A-Checker		Inglés, italiano	WCAG 1.0, section 508, Stanca Act, BITV	Single pages, Restricted pages	HTML, XHTML	HTML	
AccessColor	Para comparar contraste de colores entre fondo y fuente mediante algoritmo W3C	Inglés	WCAG 1.0	Single pages	CSS, HTML	HTML	
Accessability color wheel	Para comparar contraste de colores entre fondo y fuente mediante algoritmo W3C	Inglés	WCAG 1.0				Código abierto
AccessValeat	Analiza páginas HTML y XHTML.	Inglés	WCAG 1.0, Section 508	Single pages	HTML, XHTML	HTML, XML, EARL	Necesita instalación, no es código abierto
ART Guide	Revisa validación para la W3C y requisitos de accesibilidad por los EEUU	Inglés	WCAG 1.0, Section 508	Single pages	HTML, XHTML		Software libre
ART simulator	Simula cómo de fácil es un sitio para ser usado por personas con discapacidad	Inglés	WCAG 1.0		CSS, HTML, XHTML		Software libre
Colour Blindness	Recolorea para adaptar los colores dentro de los usados para una persona con deficiencia de color rojo/verde	Inglés	WCAG 1.0				Software libre
Colour Contrast Analyser	Para comparar el contraste de dos colores usando el algoritmo de la W3C	Portugués, Inglés, Japones y	WCAG 1.0				Software libre
CSS Analyser	Comprueba la validación de tu CSS contra los servicios de validación de la W3C	Inglés y español	WCAG 1.0	Single pages	CSS	HTML	Software libre
EvalAccess	Evalúa de accesibilidad web usando tecnología de Servicios Web	Inglés	WCAG 1.0	single pages, page groups	HTML	HTML, XML	Software libre
Flicker Rate Test for Gif Images	comprueba las imágenes de una web por su parpadeo y contraste de color, sobre todo para personas epilépticas.	Inglés, español e italiano	WCAG 1.0, section 508, Stanca Act, BITV, JIS	single pages	CSS, HTML, XHTML	HTML	Software libre
Foreground/background color contrast analyzer	calcula el contraste de color entre el fondo y el primer plano según la algoritmo de la W3C	Inglés	WCAG 1.0			HTML	Software libre
Functional Accessibility Evaluator	Analiza el código que es consistente con el uso de DRES/CITIES HTML de prácticas mejores para entornos de funcionalidad de accesibilidad web realiza algunos test W3C automatizados y luego guía al usuario acerca de lo que necesita hacer o confirmar	Inglés	WCAG 1.0, Section 508	single pages, page groups	CSS, HTML, XHTML	HTML, XML	Software libre, código abierto
Hera		Multilinguaje	WCAG 1.0	single pages	CSS, HTML, XHTML, PDF	HTML, EARL	Software libre, código abierto
HiSoftware Page Tester	Proporciona inmediatamente el estado de la calidad y accesibilidad del contenido de las páginas	Inglés	WCAG 1.0, Section 508	single pages	HTML	HTML	Software libre, comercial
HiSoftware Cynthia Says Portal	Identifica errores en el contenido relacionados con los estándares Sección 508 y el WCAG guidelines, solo valida una página cada vez	Inglés	WCAG 1.0, Section 508	single pages	HTML, XHTML	HTML	Software libre
Image analyser	examina todas las imágenes encontradas en una página web para algún punto de accesibilidad (ancho, altura, alt, ...)	Inglés	WCAG 1.0		HTML, XHTML		Software libre
Luminosity Contrast Ratio Analyser (Beta)	Permite testear las combinaciones de color de fondo y primer plano contra el contraste de luminosidad según el algoritmo http://juicystudio.com/article/luminositycontrastratioalgorithm.php	Inglés	WCAG 1.0				Software libre
Ocawa	ejecuta un test basado en la W3C usando un sistema experto empotrado	Inglés y francés	WCAG 1.0	single pages, page groups	HTML	HTML	Versión comercial o demo
Readability index calculator	calcula una puntuación de legibilidad	Danés, holandés, inglés, francés	WCAG 1.0, Section 508	single pages	CSS, HTML, XHTML	HTML	Software libre
Readability test	determina como de legible es el contenido	Inglés	WCAG 1.0		XHTML		Software libre
Relaxed HTML Validator	implementación de un validador para el cual no es necesario usar la DTD oficial de W3C, usa su propio DTD escrito en Relax NG, incluye la mayoría de especificaciones de la W3C y WCAG	Inglés	WCAG 1.0		HTML, XHTML, SVG	XML	Código abierto
SiteMorse	ofrece un ranto de servicios de sitios web de test que no requiere instalación, descargas o soporte técnico para operar	Inglés	WCAG 1.0	Single pages, page groups, restricted	HTML, XHTML	HTML	Versión comercial o demo
TAW Online	Web accessibility Test, incluye realimentación	Inglés, español	WCAG 1.0 Y 2.0	single pages	HTML, XHTML	HTML	Software libre
Tidy	validador fácilmente incorporable a otros software	Inglés	WCAG 1.0	single pages	HTML, XHTML		Software libre, código abierto
Torquemada	ofrece una completa metodología para el análisis de accesibilidad por páginas	Inglés, italiano	WCAG 1.0	single pages	CSS, HTML, XHTML	HTML	
Total validator	herramienta de validación, validador de comprensión de HTML, validador de accesibilidad, verificador de "encanto", validador de enlaces rotos, habilidad para tomar capturas de pantalla con diferentes navegadores en diferentes plataformas	Inglés	WCAG 1.0, Section 508	Single pages, page groups, restricted pages	HTML, XHTML	HTML	Software libre, comercial
Vischeck	Muestra como se ven las cosas para una persona ciega.	Inglés, japonés	WCAG 1.0, BITV		CSS, HTML		Software libre, código abierto
WAEX	Un fichero XSLT es aplicado para generar un informe de accesibilidad	Inglés	WCAG 1.0	Single pages	XHTML	HTML, XML, EARL	Software libre, código abierto
Wahelper	es un conjunto sw de herramientas extensibles para la validación semiautomática de la accesibilidad web.	Inglés	WCAG 1.0, Section 508	single pages, page groups	CSS, HTML, XHTML	HTML	Software libre, código abierto
WAVE	da errores para los que se necesita de la decisión humana	Inglés	WCAG 1.0, Section 508	Single pages	CSS, HTML, XHTML	HTML, XML, EARL	Software libre, código abierto
Webagogo	Evalúa la calidad de los sitios en cuanto a: accesibilidad, usabilidad, velocidad, SEO,...	Inglés	WCAG 1.0, Section 508, Stanca Act, BITV	Single pages, page groups	HTML, XHTML	HTML	Software libre, código abierto
WebXACT	Valida el contenido de los sitios web por accesibilidad, calidad y privacidad	Inglés	WCAG 1.0, Section 508	Single pages	HTML, XHTML	HTML	Software libre

Tabla 5 Comparativas de los principales validadores Web

Para cada uno de los validadores interesa recoger las características siguientes:

- Idiomas en los que está disponible.
- Directrices que es capaz de validar.
- Si es capaz de comprobar únicamente la página que se le pasa como entrada o, si por el contrario, es capaz de realizar el mismo proceso sobre otras a las que se llega mediante los hipervínculos en ésta.
- Los formatos que acepta para la página de entrada al programa.
- El formato de la salida, las observaciones sobre el modo en el que está disponible como software libre o si es de código abierto.

Tras esta clasificación, en las siguientes etapas, únicamente se prestará atención a aquellos validadores que están disponibles en Web y que además no necesitan de una instalación previa.

Como conclusiones importantes de los datos recogidos, es que la mejor validación se ofrece con el conjunto de directrices WCAG 1.0 [4], section 508, Stanca. Act y BITV. Algunos de los sistemas son capaces de ejecutar una validación de forma recursiva, lo que se nombra en la tabla como “page groups”. Esta será una de las opciones de mejora que se tendrán en cuenta ya que WAEX [1] no lo incorpora y es uno de los objetivos posibles.

Para el caso de los formatos de las páginas de entrada al sistema, se tendrán en cuenta todos ellos, de modo que el validador pueda aceptar cualquier página HTML [7] o XHTML [8] y se tendrá en cuenta también su hoja de estilo en formato CSS [9].

Para completar la información que se dispone de cada uno de los validadores recogidos anteriormente, se ha recogido un resumen de aquellas características que se han encontrado en ellos y que realizan una validación sobre una regla que WAEX [1] no hace. Tras ser anotados todos los errores y comprobarlos con los que proporciona WAEX [1] se encuentran las siguientes posibles mejoras que actualmente WAEX [1] no es capaz de reconocer:

Comprobaciones automáticas

Las comprobaciones automáticas son aquellas que no dependen del contexto o contenido de los elementos de la página, únicamente de la estructura.

- Algunos validadores comprueban el uso de combinaciones de teclas como atajos a las funcionalidades más importantes y otras que ayudan a la navegación entre los distintos elementos de una página.

Comprobaciones semiautomáticas

En cuanto a las comprobaciones semiautomáticas, son aquellas que indican un posible error, y queda a responsabilidad del desarrollador examinarlo y decidir si es necesario modificarlo.

De estas comprobaciones, se han obtenido algunas reglas que deben ser aplicadas en WAEX [1]:

- El texto alternativo, indicado por el atributo alt en los elementos de imágenes no debe ser demasiado largo. Es una regla que puede ser fácilmente detectable si se encuentra un texto alternativo mayor que una longitud definida. El problema, y lo que hace que esta regla sea por tanto semiautomática es que no se conoce a priori el valor de esta longitud en la que el texto pasa a ser categorizado como “demasiado largo”. Actualmente, WAEX [1] permite al usuario definir cuál debe ser esta longitud máxima, por tanto, aunque no forme parte de la regla se mantendrá esta filosofía en el nuevo validador.
- Algunos de los validadores que se han estudiado, su objetivo no es ceñirse a las validaciones automáticas de la estructura y datos de cada página Web, sino que pretenden dar información acerca del aspecto que presentan, concretamente en cuanto a los colores utilizados. La forma de decidir si dos colores tomados uno como fondo y otro como el color de la fuente son apropiados en su combinación parece bastante trivial ya que la W3C [3] nos proporciona dos fórmulas cuyo resultado da esta información.

Estas fórmulas están publicadas en la regla 1.4 de la W3C [3]. En ella se dice que la presentación visual de imágenes y texto deben estar en una razón de al menos 4.5:1. De esta regla se excluye el texto con una fuente de gran tamaño, texto decorativo y logotipos. Para decidir si se cumple esta razón en dos colores A y B de los que se conoce sus valores RGB (Red, Green, Blue) se aplican primero el cálculo del brillo para cada uno de ellos (ver Tabla 6):

$$\text{Brillo_color_A} = ((\text{Red_value_color_A} \times 299) + (\text{Green_value_color_A} \times 587) + (\text{Blue_value_color_A} \times 114)) / 1000$$

Tabla 6 Fórmula para el cálculo de diferencia de brillo

Este resultado del valor absoluto de la diferencia del cálculo anterior para el color A y para el color B deber ser mayor que 125 para ser considerados como valores correctos respecto de la diferencia de brillo.

Por otro lado, la diferencia de color permitirá también que los objetos a los que se les aplique estos colores sean legibles. Se expresa mediante la siguiente fórmula (Tabla 7):

$$\text{Diferencia_de_color} = (\text{maximum}(\text{Red value 1}, \text{Red value 2}) - \text{minimum}(\text{Red value 1}, \text{Red value 2})) + (\text{maximum}(\text{Green value 1}, \text{Green value 2}) - \text{minimum}(\text{Green value 1}, \text{Green value 2})) + (\text{maximum}(\text{Blue value 1}, \text{Blue value 2}) - \text{minimum}(\text{Blue value 1}, \text{Blue value 2}))$$

Tabla 7 Fórmula para el cálculo de la diferencia de contraste

En este caso, dos colores serán considerados como válidos en este aspecto si su diferencia de color es un valor mayor que 500. Validación de color (regla 2.2) [10].

Ya que se considera que la validación de color completaría el resultado proporcionado por WAEX [1], se ha decidido tomar esta regla como un requisito del proyecto.

Éstas son todas las características relevantes que se han conseguido obtener de los validadores que hay disponibles en el Web.

3.2. Selección de los objetivos

Este punto del documento se centra en la selección de los objetivos formulados como las mejoras que se han decidido aplicar a WAEX [1]. Esta sección es un complemento al punto anterior de objetivos. Servirá para ayudar a la definición de los requisitos en los siguientes puntos y tener un esquema claro de los pasos que se siguieron en el desarrollo.

Estos objetivos han sido seleccionados a partir del estudio anterior o por petición expresa del tutor del proyecto. Una vez formulados como requisitos, que se listan a continuación, fueron aprobados por Vicente Luque, tutor de este proyecto. La selección de los objetivos se ha realizado a partir de las posibles mejoras que se consideraban viables y más necesarias. A lo largo del proyecto se describe el proceso de cada uno de estos objetivos junto con los problemas y soluciones, a pesar de los cuales se ha conseguido resolver y obtener una solución conforme a la propuesta inicial.

En la definición de los objetivos, a cada uno de ellos se le ha asignado una prioridad, que ha sido seguida en el desarrollo e implementación. Esta prioridad es definida mediante un valor numérico para cada uno de ellos, y se considerará más importante aquel objetivo que tenga menor valor de prioridad. Así, encontramos los siguientes objetivos del sistema, organizados en la Tabla 8:

Prioridad	Objetivo	Descripción	Procedencia
1	Software dependiente de WAEX	Todos los cambios para la nueva aplicación serán incorporados como una capa que dependerá de WAEX pero no lo modifique.	A petición de Vicente Luque Centeno.
1	Validación recursiva	Se validarán todas las páginas a las que se pueda llegar a través de la navegación por los hipervínculos de la página de entrada y sucesivas.	Mejora procedente de la comparativa con los otros validadores.
2	Soporte a cookies	Se dará soporte a validación de aquellas páginas que precisen de acceso mediante cookies.	A petición de Vicente Luque Centeno.
3	Soporte a cambios online	Se proporcionará una plataforma para realizar aquellos cambios para los errores que se consideran posibles online.	A petición de Vicente Luque Centeno.
4	Validación de color	Validación de los colores empleados en cada una de las páginas.	Mejora procedente de la comparativa con los otros validadores.
5	Mejorar hoja de estilos	Realizar cambios en las páginas para aplicar más elementos de estilo, en particular para cambios en color y elementos HTML [7] obsoletos.	Procedente por iniciativa propia.

Tabla 8 Definición final y detallada de los objetivos

3.3. Casos de uso

En este punto se incorpora un tipo de diagrama UML, los “Casos de uso” con el fin de completar el análisis de los objetivos, previo a una recogida de los requisitos. Con esto, se pretende clarificar qué será capaz de hacer el sistema final y qué papel tendrán los actores en su funcionamiento.

Ya que el nuevo sistema será una capa que funcionará por encima de WAEX [1], se considerará a éste como un sistema ya creado y al que se le realizarán peticiones. Aunque esto está representado en los casos de uso, las llamadas a WAEX [1] quedarán transparentes al usuario.

Inicialmente, el sistema hará un recorrido por los hipervínculos de la página de entrada para obtener el conjunto de todas ellas que podrán ser validadas. Es por esto por lo que se diferencian dos tipos de usuarios, el usuario inicial y el usuario completo. Se denomina usuario inicial aquel que entra en la aplicación y envía la petición de validación de una página. Una vez se realiza el recorrido de todos los enlaces, el usuario pasa a ser un usuario completo. El usuario completo realizará todas las funciones sobre cualquiera de las páginas del que se ha registrado en el paso anterior. Por lo tanto, se tendrá el escenario inicial siguiente:

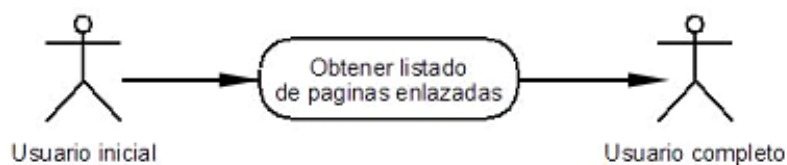


Figura 3 CU-01 Registro de páginas enlazadas a partir de la inicial

CU-01: Registro de páginas enlazadas a partir de la inicial

Nombre	Obtener listado de páginas enlazadas
Autor	Sonia Almodóvar
Fecha	20/06/2009
Actores	Usuario inicial
Descripción	El usuario inicial hace una petición de validación de una primera página que se considerará raíz. El sistema obtendrá el conjunto de páginas a partir de ésta, a las que se llega por navegación entre hipervínculos.
Precondiciones	Ninguna
Postcondiciones	El usuario pasa a ser completo ya que dispone del conjunto de páginas enlazadas y podrá optar a las opciones de los demás casos de uso.

Tabla 9 CU-01 Registro de páginas enlazadas a partir de la inicial

Escenario 2: Funcionalidad completa

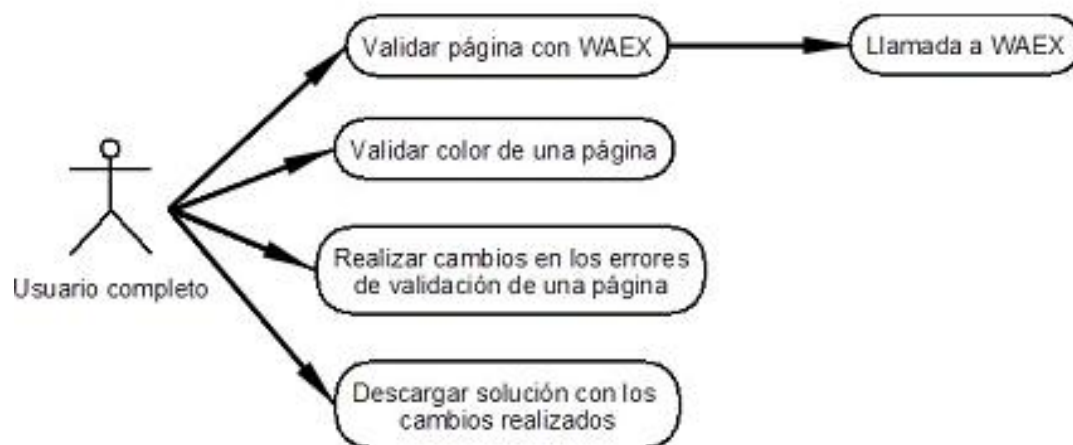


Figura 4 CU-01 de la funcionalidad completa

CU-02: Validar página con WAEX [1]

Nombre	Validar página con WAEX [1]
Autor	Sonia Almodóvar
Fecha	20/06/2009
Actores	Usuario completo
Descripción	Envía al sistema la petición de la validación de una de las páginas que están dentro del conjunto de enlazadas, para obtener el resultado de WAEX [1].
Precondiciones	Haber realizado la búsqueda de páginas enlazadas. CU-01
Postcondiciones	Ninguna

Tabla 10 CU-02: Validar página con WAEX

CU-03: Validar color utilizado en una página

Nombre	Validar color de una página
Autor	Sonia Almodóvar
Fecha	20/06/2009
Actores	Usuario completo
Descripción	Envía la petición de validación de los elementos de color de una página para que el sistema reporte los errores encontrados.
Precondiciones	Haber realizado la búsqueda de páginas enlazadas. CU-01
Postcondiciones	Ninguna

Tabla 11 CU-03: Validar color utilizado en una página

CU-04: Facilitar cambios en el momento en páginas para solucionar errores de validación.

Nombre	Realizar cambios en los errores de validación de una página
Autor	Sonia Almodóvar
Fecha	20/06/2009
Actores	Usuario completo
Descripción	Realiza los cambios oportunos sobre una página para resolver los problemas de validación a partir de una interfaz que facilita estos cambios.
Precondiciones	Haber realizado la búsqueda de páginas enlazadas. CU-01
Postcondiciones	Ninguna

Tabla 12 CU-04: Facilitar cambios en los errores de validación.

CU-05: Descarga de la solución con los cambios realizados.

Nombre	Descargar solución con los cambios realizados.
Autor	Sonia Almodóvar
Fecha	20/06/2009
Actores	Usuario completo
Descripción	Hace una petición al sistema para que éste devuelva el código de todas las páginas que han sido validadas y corregidas.
Precondiciones	Haber realizado la búsqueda de páginas enlazadas. CU-01
Postcondiciones	Ninguna

Tabla 13 CU-05: Descarga de la solución con los cambios realizados.

Estos casos de uso recogen la funcionalidad de WAEX2. En los capítulos siguientes se detallará la recogida de todos los requisitos que son necesarios para un sistema que cumpla con estas funcionalidades y su posterior diseño y desarrollo.

4. CAPÍTULO III: ANÁLISIS

4.1. Alcance del sistema

Actualmente WAEX [1] es una herramienta de validación Web. El desarrollo de la capa complementaria que aumentará la funcionalidad será capaz de cumplir las siguientes especificaciones:

La aplicación debe seguir soportando la funcionalidad del sistema actual. Es decir, el desarrollo de la nueva aplicación, no influirá en el funcionamiento de la anterior.

WAEX2 será capaz de validar todas las páginas a las que se pueda acceder mediante navegación entre los hipervínculos de la página proporcionada y las siguientes obtenidas.

WAEX2 sólo seleccionará de los hipervínculos encontrados en las páginas a validar, aquellos cuya URL pertenezca al mismo dominio que la página de entrada o página inicio. Con esto se pretende que sólo se validen páginas pertenecientes a un mismo sitio. La validación se realizará siguiendo las reglas WCAG 1.0 [4] publicadas por la W3C [3].

WAEX2 incorporará un validador de color. Su función será identificar todos los puntos posibles en el documento en los que el uso de un color de letra y un color de fondo no sean apropiados por no cumplir las reglas de validación. En concreto, la regla 2.2 [10] de las WCAG 1.0 [4]. Este punto está sometido a ciertas limitaciones por las que no estará capacitado para comprobar el correcto uso de color en el texto cuyo fondo es una imagen. Se debe a que para resolver el problema, sería necesario hacer un estudio riguroso del aspecto de la página, donde se sitúa cada elemento y un análisis de los colores de la imagen en cada píxel. Además, debido a que el aspecto de la página puede cambiar dependiendo del estilo, los colores de fondo y de fuente que se tomarán como referencia para ser analizados, serán únicamente aquellos que son aplicados a bloques.

Con esto se evita que si un bloque está en una posición de la página que influye en el color de un elemento cuyo código no es un subelemento del anterior, no se tendrá en cuenta. En los casos en los que varios subelementos tienen propiedades de color distintas, y debido a que el CSS [9] se aplica a modo cascada, serán analizadas todas las combinaciones de colores posibles entre los colores de fondo y los de fuente, para subelementos. El método de valoración de un uso correcto de color, será aplicando las dos siguientes fórmulas a cada par de colores, y testeando con el umbral que se proporciona. Para que un par de colores utilizados sea correcto debe superar dicho umbral:

$$\text{Brillo_color_A} = ((\text{Red_value_color_A} \times 299) + (\text{Green_value_color_A} \times 587) + (\text{Blue_value_color_A} \times 114)) / 1000$$

$$\text{Diferencia_de_color} = (\text{maximum (Red value 1, Red value 2)} - \text{minimum (Red value 1, Red value 2)}) + (\text{maximum (Green value 1, Green value 2)} - \text{minimum (Green value 1, Green value 2)}) + (\text{maximum (Blue value 1, Blue value 2)} - \text{minimum (Blue value 1, Blue value 2)})$$

WAEX 2 estará disponible en Web o como aplicación complemento de Mozilla Firefox.

WAEX 2 tendrá soporte a cookies. Será capaz de validar todas las páginas a través de los hipervínculos de una inicialmente dada, y accediendo a ellas aun con petición de cookies por parte del servidor que las proporciona. Estas cookies serán el conjunto de las almacenadas en el cliente en el momento de la petición de validación de la primera página. Por problemas de seguridad y recogida de estas cookies, puesto que no fueron creadas por el mismo servidor que realiza la validación, esta funcionalidad sólo estará disponible en los casos en que se acceda a la aplicación desde la extensión de Mozilla Firefox.

Por último, WAEX 2 será además una herramienta innovadora dentro del campo de los validadores existentes. Se dará soporte para la post-validación, en el que se podrá cambiar en la misma página de la validación aquellos errores que reporta con el fin de

cambiar en el mismo momento y no tener necesidad de acceso al código por parte de los desarrolladores para solucionar el problema. Como aclaración, en cuanto a los cambios, debido a que no se tiene acceso sobre el servidor donde se encuentran las páginas, se realizarán sobre una copia del documento y se almacenará de forma temporal en el servidor de la aplicación. Tras la validación el usuario tendrá la opción de descargar el código con los cambios ya realizados. Esta funcionalidad solo será posible para aquellas reglas automáticas que no dependan de la semántica y no implique el cambio de gran parte de la estructura del HTML [7] o XHTML [8]. Las reglas que se han decidido incorporar a la post-validación son las siguientes:

- Regla 1.1 [11] Para todos los elementos no textuales que proporcionan algún tipo de información deben ir acompañados del atributo ALT cuyo valor será una descripción detallada de la información visible en el elemento. Además esta descripción no podrá ser una cadena vacía.
- Regla 1.2 [11] Proporcionar enlaces de texto redundantes para cada región activa de un mapa de imágenes en el lado del servidor.
- Regla 2.2 [12] Asegurar que todas las combinaciones de colores de fondo y fuente tienen suficiente contraste cuando son visualizados por personas con algún tipo de discapacidad relacionada con la distinción de colores o cuando son visualizadas en pantallas en dispositivos con limitación de colores.
- Regla 4.3 [13] Todos los documentos HTML [7] deben disponer de información acerca del lenguaje natural del documento.
- Regla 5.5 [14] Al igual que ocurre con las imágenes, otro elemento problemático a la hora de recoger la información que proporciona, son los elementos utilizados para representar tablas. Todas las tablas de un documento, encabezadas por <TABLE> deben tener el atributo *summary* cuyo valor sea una descripción de la información disponible en la tabla. Además, este valor no puede ser una cadena vacía.

- Regla 12.1 [15] Título para cada elemento FRAME para facilitar la identificación y navegación entre estos elementos.
- Regla 13.1 [16] Todos los documentos HTML [7] deben tener el elemento TITLE en la cabecera. Su cometido es dar información muy resumida acerca de lo que se va a ver en el documento. Este valor no puede ser una cadena vacía.
- FONT: Los elementos FONT, son elementos que han quedado obsoletos debido a que no cumplen con la separación de la información de la estructura en la que se muestra en pantalla. La aplicación resolverá este problema cambiando todos los elementos FONT por elementos SPAN asociados a un estilo propio con las mismas características que tenía el anterior de forma que se preserve la estructura y estilo del documento.

4.2. Identificación de los interesados del sistema

Los interesados en el sistema son personas físicas y organizaciones que están involucradas activamente en el proyecto, o cuyos intereses puede ser positiva o negativamente afectados como resultado de la ejecución del proyecto o de sus resultados.

El cliente es la persona u organización que pide el desarrollo del sistema. En general, no tiene por qué coincidir con el usuario final, aunque en este caso podría decirse que el cliente pide el sistema para usarlo en su propia organización. Se trata de la Universidad Carlos III de Madrid, aunque el usuario del sistema será la persona que utilice la aplicación directamente, pudiendo ser de la misma universidad o externa si se decide hacer público este validador.

Es una aplicación que tiene como objetivo validar páginas HTML [7] o XHTML [8]. La entrada del sistema es únicamente la URL de la página que se desea validar. Para que el usuario sea capaz de tratar la información y modificar la página que ha sido validada se requieren conocimientos de programación y diseño de sitios WEB, así como del conocimiento de las WCAG [11] en cuanto a sus respuestas y su finalidad. WAEX [1] es por tanto, una herramienta para garantizar el correcto diseño de un sitio a partir de la validación de cada una de las páginas que lo componen.

Dado que la nueva aplicación que se desea desarrollar es una ampliación y mejora de las funcionalidades de WAEX [1], sus posibles usuarios serán los mismos que para la actual. Estos usuarios serán aquellos desarrolladores que deseen garantizar que sus páginas son accesibles a todos los usuarios finales. Estos desarrolladores tendrán conocimiento de la sintaxis de HTML [7] así como la posibilidad de cambiar el código de sus propias páginas.

4.3. Identificación y definición de los requisitos del sistema

Los Requisitos Software tienen como misión determinar la funcionalidad del producto. Estos Requisitos son una referencia que se deberá de corroborar también una vez realizado el diseño y el producto final. Los requisitos software no se obtienen a partir de los casos de uso sino que su adquisición ha sido un proceso paralelo a éstos de modo que se obtenga la máxima información acerca de qué se quiere obtener.

Antes de proceder a su redacción se debe establecer un formato de tabla que se utilizará para albergar dichos requisitos. Dicha tabla contendrá los siguientes campos:

- **Identificador:** cada requisito software deberá estar identificado de forma unívoca con el objetivo de facilitar su trazabilidad en las fases subsiguientes, si se considerase necesario, o hacer referencia a éste. El formato de ficho identificador será RSX-NN. NN indica el número de requisito de ese tipo. El significado de RS es Requisito Software y X indica el tipo de requisito según la Tabla 14.

VALOR DE X	SIGNIFICADO
F	Funcional
I	Interfaz
O	Operacional
R	Rendimiento
SF	Seguridad ante fallos
C	Calidad
S	Seguridad ante amenazas externas
D	Documentación
In	Inversos

Tabla 14 Descripción de la nomenclatura de requisitos

- Necesidad: Todo requisito puede ser esencial, deseable u opcional, los requisitos esenciales se deberán cumplir a rajatabla y no son negociables. Los otros dos pueden verse sujetos a modificaciones.
- Prioridad: Cada requisito debe tener establecida una prioridad que sirve al desarrollador de referencia a la hora de planificar la creación del sistema.
- Estabilidad: Algunos requisitos pueden presentarse fijos a lo largo de toda la vida del proyecto mientras que otros dependen de otros factores y pueden estar sujetos a cambios.
- Fuente: Todo requisito software puede estar relacionado con uno o varios requisitos de usuario de los que parten.
- Descripción: Describe de forma clara, verificable y concisa el requisito.

A continuación se muestra la tabla de requisitos que se usará como plantilla (Tabla 15):

Identificador: RSX-NN	
Prioridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente:
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	
Descripción:	

Tabla 15 Descripción del estilo de tabla para la definición de los requisitos

Los requisitos software se clasificarán en:

- Funcionales: Describen el ‘qué’ debe de hacer el software.
- Interfaz: Referente a la interfaz de comunicación que presentará la aplicación y que permitirá a ésta comunicarse con el usuario.
- Rendimiento: Requisitos que garanticen el buen funcionamiento de la aplicación.

- Operacionales: Requisitos que indiquen el entorno de ejecución de la aplicación.
- Documentación: Aquellos requisitos que estén relacionados con temas de ayudas y manuales de usuario.
- Seguridad ante amenazas: Requisitos que garanticen la seguridad de la aplicación ante posibles ataques tanto internos como externos.
- Portabilidad: Requisitos que doten a la aplicación de flexibilidad ante cualquier plataforma de ejecución.
- Calidad: Son aquellos requisitos que garantizan un producto de calidad.
- Seguridad ante Fallos: Requisitos que permiten a la aplicación recuperarse ante fallos u operaciones indebidas realizadas por los usuarios.
- Inversos: Permite aclarar y puntualizar factores no considerados por la aplicación.

4.3.1. Requisitos funcionales

Identificador: RSF-01	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: CU-01
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Obtener un listado de todas las páginas a las que se puede acceder a través de los hipervínculos de una página inicial.

Tabla 16 RSF-01 Listado de todas las páginas enlazadas

Identificador: RSF-02	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: CU-02
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja <input type="checkbox"/>	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Permite que una página de entrada, seleccionada por el usuario o las derivadas de ésta, sea validada por WAEX.

Tabla 17 RSF-02 Validación de una página por WAEX

Identificador: RSF-03	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: CU-03
Necesidad: <input type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input checked="" type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Permite que una página de entrada, seleccionada por el usuario o las derivadas de ésta, sea validada por las reglas de uso de color.

Tabla 18 RSF-03 Validación del color de una página

Identificador: RSF-04	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente) y CU-04
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Solo podrá ser aplicable a aquellas reglas que puedan cambiarse mediante una entrada del usuario y sean cambios referentes a reglas de comprobaciones automáticas.
Descripción:	Permitir realizar cambios en una página, sin necesidad de acceder al código, para solucionar problemas de validación.

Tabla 19 RSF-04 Permitir cambios post-validación en WAEX

Identificador: RSF-05	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: CU-05
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Permite al usuario descargar sus propias páginas una vez se hayan sometido a las correcciones oportunas para resolver los problemas de validación.

Tabla 20 RSF-05 Descarga de las páginas validadas y corregidas

Identificador: RSF-06	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: CU-05
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Para facilitar el trabajo del desarrollador, la descarga de la solución según el requisito RSF-05 se realizará siguiendo el mismo esquema de carpetas en el que están dichas páginas almacenadas en el servidor de origen.

Tabla 21 RSF-06 Almacenamiento y descarga de la solución

4.3.2. Requisitos de interfaz

Identificador: RSI-01	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Sonia Almodóvar
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Los dos tipos de validación que existirán serán separados en dos interfaces distintas, pues su funcionalidad es diferente.

Tabla 22 RSI-01 Dos tipos de interfaces para los dos resultados de validación.

Identificador: RSI-02	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Sonia Almodóvar
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	A través de la interfaz, será posible seleccionar qué página será validada de entre todas las que componen el listado que se creó inicialmente, según el RSF-01

Tabla 23 RSI-02 La interfaz permite seleccionar qué página validar.

Identificador: RSI-03	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente)
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	La interfaz inicial debe ser similar a la de WAEX [1], proporcionando exactamente las mismas entradas de datos de forma que no sea necesario un aprendizaje previo por parte del usuario para poder utilizar la nueva herramienta.

Tabla 24 RSI-03 Interfaz de entrada con los mismos datos de entrada que WAEX

Identificador: RSI-04	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente)
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Existirán dos interfaces de entrada a la aplicación, una que se lance desde una extensión de Mozilla Firefox y la otra desde una página HTML [7]. Ambas llevarán a la misma página de validación donde comienza la ejecución de la aplicación.

Tabla 25 RSI-04 Interfaces de entrada al sistema

Identificador: RSI-05	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Sonia Almodóvar
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Todas las funcionalidades podrán ser accedidas directamente desde la interfaz de validación.

Tabla 26 RSI-05 Interfaz de validación con todas las funcionalidades

Identificador: RSI-06	
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Sonia Almodóvar
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	El espacio de pantalla de la página seguirá los estándares definidos por la W3C [3]

Tabla 27 RSI-06 Espacio de página restringido por el estándar W3C

Identificador: RSI-07						
Prioridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Sonia Almodóvar					
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional						
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja					
Estabilidad:	Estable					
Descripción:	<div>La interfaz de validación deberá tener un aspecto según la siguiente estructura:</div> <table><tr><th>Tipo de validación y página validada</th><th>Funcionalidades</th></tr><tr><td rowspan="2">Resultado de la validación</td><td></td></tr><tr><td>Detalles de la validación</td></tr></table> <div>Donde el tipo de validación diferenciará entre color o la realizada por la antigua aplicación WAEX [1] y la página validada es la ruta de la página para la que se está mostrando el resultado de validación.</div>	Tipo de validación y página validada	Funcionalidades	Resultado de la validación		Detalles de la validación
Tipo de validación y página validada	Funcionalidades					
Resultado de la validación						
	Detalles de la validación					

Tabla 28 RSI-07 Estructura de la interfaz de validación.

4.3.3. Requisitos de rendimiento

Identificador: RSR-01	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente)
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Depende del tiempo de respuesta de WAEX [1]
Descripción:	Será necesario el desarrollo de una aplicación que no tenga altos tiempos de respuesta a peticiones de usuario. La máxima permitida será para la validación por tipo WAEX [1] cuyo tiempo dependerá del sistema anterior, aproximadamente un máximo de 30 segundos. Para el resto de funcionalidades no deberá exceder los 10 segundos de espera.

Tabla 29 RSR-01 Reducir al máximo el tiempo de respuesta de WAEX.

4.3.4. Requisitos operacionales

Identificador: RSO-01	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente)
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	La aplicación podrá ser ejecutada desde una extensión de Mozilla Firefox o directamente desde una página HTML [7].

Tabla 30 RSO-01 Modos de ejecución de la aplicación

Identificador: RSO-02	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente)
Necesidad: <input checked="" type="checkbox"/> Esencial <input type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	La aplicación será ejecutada en un servidor con sistema operativo Linux.

Tabla 31 RSO-02 El sistema funciona en un servidor con sistema operativo Linux.

4.3.5. Requisitos de documentación

Identificador: RSD-01	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente)
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	La aplicación irá acompañada de un manual de usuario en formato digital. Este manual formará parte del documento descriptivo del proyecto.

Tabla 32 RSD-01 Manual de usuario

4.3.6. Requisitos de seguridad

No se considera necesario prestar atención a distintos perfiles de usuario, ni ningún tipo de seguridad puesto que no compromete datos personales ni necesarios de una restricción de acceso ya que todos los datos que se obtienen a través del programa ya han pasado sus propios medios de seguridad, en los casos necesarios.

4.3.7. Requisitos de portabilidad

Identificador: RSP-01	
Prioridad: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja	Fuente: Tutor del proyecto (cliente)
Necesidad: <input type="checkbox"/> Esencial <input checked="" type="checkbox"/> Deseable <input type="checkbox"/> Opcional	
Claridad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja	Verificabilidad: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Estabilidad:	Estable
Descripción:	Se podrá acceder al sistema (sin funcionalidad de soporte a cookies) desde cualquier navegador a la página inicio de WAEX 2.

Tabla 33 RSP-01 Acceso al sistema desde cualquier navegador.

4.4. Identificación de las alternativas

Para el desarrollo del proyecto se considera que las principales funcionalidades deberán ser módulos separados. Esta decisión se ha tomado porque de esta forma, mediante modulación, se harán independientes cada una de las partes del sistema, simulando la existencia de distintas capas por encima de WAEX [1] y no solo una. Además, esta decisión proporciona mayor fiabilidad frente a los cambios que puedan producirse en una capa, los cuáles no afectarán al resto.

Por lo tanto, se definen los siguientes módulos de la aplicación:

- Entrada al sistema
- Búsqueda páginas
- Validación WAEX y validación de color
- Modificación de páginas
- Salida del sistema

4.4.1. Entrada al sistema

Para el módulo de entrada al sistema no era necesario aplicar ningún tipo de decisión, únicamente seguir las reglas impuestas por los requisitos. La interfaz y entradas serán similares a la actual interfaz de WAEX [1]. En el caso de la extensión, se intentará hacer lo más sencilla posible, es decir, que necesite de los mínimos pasos a través de ella para acceder a la aplicación WAEX [1] y manteniendo las entradas al sistema similares a WAEX [1]. Estas entradas son el campo de entrada de la URL de la página a validar, y las opciones de validación de ésta: máxima longitud del atributo *alt*, *title*, máximo número de campos en los formularios, etc. Es decir, debería ser similar a la Figura 5:

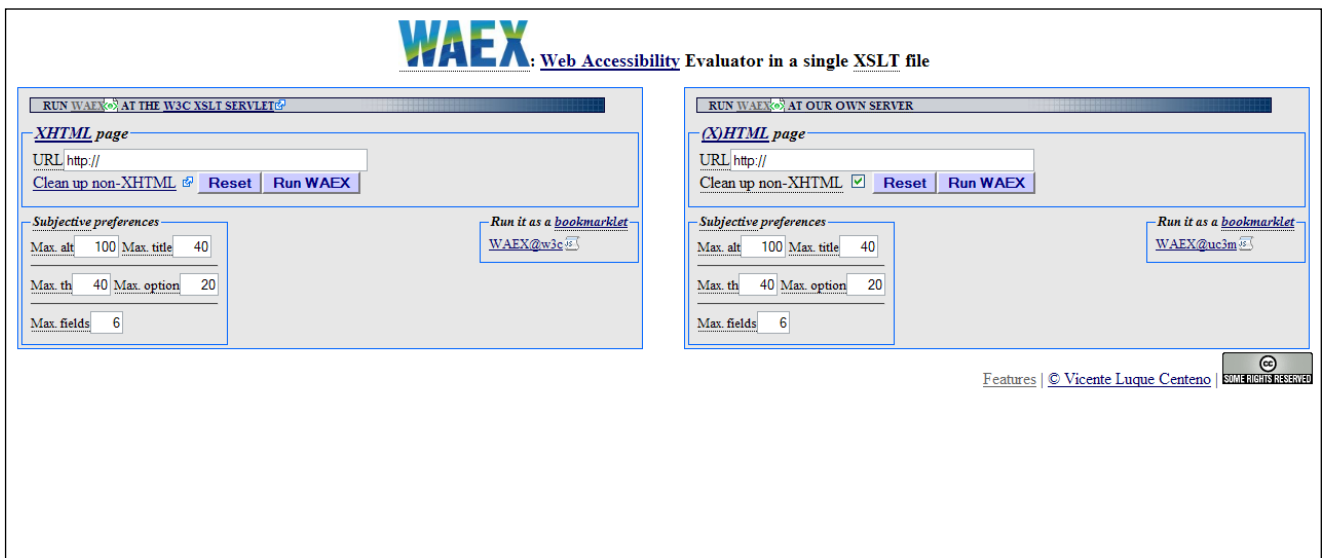


Figura 5 Interfaz de entrada a la aplicación actual WAEX.

4.4.2. Recorrido de páginas

Para el recorrido en búsqueda de las distintas páginas se consideran varias opciones a tener en cuenta. Primero la forma de petición de las páginas vía web. Puesto que la aplicación estará instalada en un servidor Linux, los comandos posibles son GET [17] y WGET [18]. La diferencia entre ellos reside en las opciones que permiten. GET [17] es un comando más simple y el resultado de la página es más fácil de tratar puesto que la proporciona por la salida estándar. En el caso de WGET [18] permite enviar cookies y peticiones con datos por vía post, pero el tratamiento del resultado es un poco más trabajoso debido a que la salida es la ruta del fichero donde se guarda la información de la petición.

Para el recorrido de páginas en busca de enlaces se plantean dos opciones. Una de las posibilidades sería recorrer el código como texto plano y hacer búsquedas del patrón que indica el inicio de elementos que pueden llevar a otras páginas, como son los enlaces, las imágenes, áreas y frames. Se analiza el código de este elemento y se obtiene el valor del atributo que contiene el enlace. La segunda opción es utilizar una librería para manejar la página como una estructura árbol en la que los nodos son los elementos HTML [7]. De esta última forma se simplificaría la búsqueda.

4.4.3. Validación de WAEX y validación de color

La validación de WAEX [1] puede obtenerse de dos formas. Una posibilidad es realizar una petición de la misma al igual que lo hace WAEX [1], que resulta algo muy sencillo ya que los datos son pasados vía GET¹, es decir, en la misma URL. Las ventajas de esta petición es que el nuevo sistema conservaría el requisito de ser una capa independiente de WAEX [1].

La segunda opción, es realizar una validación teniendo una copia del fichero XSLT que utiliza WAEX [1]. La funcionalidad y el resultado son los mismos con la ventaja añadida que si en un futuro se decide realizar algún cambio en WAEX [1], también sería necesario actualizar este fichero, por lo que ya no se tendría una capa independiente de la forma en la que funciona por dentro WAEX [1], sino una copia de éste.

Para el caso de la validación de color, existe la opción de implementar el algoritmo para conocer los posibles errores, o utilizar un sistema ya existente. En el primer caso, implicaría resolver problemas acerca de cómo decidir qué pares de colores se debe someter a validación, y lo que ocurre al tener hojas de estilos aplicadas a elementos en cascada. Por el otro lado, el validador de este tipo más eficiente que se ha encontrado es AccessColor, disponible en la URL que se indica en la referencia [19]. Este validador devuelve el resultado de la validación de color en una página HTML [7]. La forma de acceder a esta segunda opción sería mediante una petición vía POST, por lo que sería necesario el uso del comando WGET [18]. La ventaja de realizar las peticiones frente a implementar un nuevo sistema es que se evitan muchos problemas que ya han sido resueltos, como seleccionar qué colores deben ser enfrentados, y esto proporciona tiempo extra para resolver otros problemas más importantes y prioritarios del proyecto. La desventaja es que se crea una dependencia frente a cambios en esta aplicación. La decisión de optar por esta solución, es que se considera que no es una herramienta con posibles futuras variaciones en su funcionamiento, ya que no deja lugar a una ampliación de funcionalidad.

¹ Referido al método empleado en HTML para el envío de los datos de un formulario y no al comando de Linux.

4.4.4. Soporte a cookies

Éste es sin duda el mayor problema al que se enfrenta este proyecto. Las opciones para proporcionar soporte a cookies son varias y todas ellas fueron estudiadas en profundidad.

La primera opción y más sencilla sería validar todas las páginas de forma recursiva y almacenar por separado las URLs de aquellas cuyo resultado es vacío o inválido (respuesta con códigos 401 del servidor, acceso no autorizado,...). Para dichas páginas se le daría la opción al usuario de proporcionar un identificador y contraseña válidos para el acceso o bien omitir este acceso. Una vez se dispone de estos datos, se vuelve a comenzar la validación de estas páginas restantes y sus sucesivas y en aquellas en las que se encuentra error de acceso, volver a intentarlo con cada uno de los pares usuarios-contraseñas recogidos anteriormente hasta dar con uno válido. El problema de esta solución es que es un proceso muy lento y sólo una pequeña parte de las páginas serían capaces de funcionar con este formato, debido a que son redireccionadas a través de javascript [20]. El problema del javascript [20] es que no es accesible y muchas de las páginas que tienen validación, tras aceptar la identificación del usuario, realizan una redirección a la página pedida a partir de javascript [20]. Puesto que el javascript no es accesible y no debe ser utilizado, no se contemplarán estos casos.

En cuanto a la forma de proporcionar la petición de datos de nombre de usuario y contraseña en las peticiones se debería preguntar al usuario cómo debe realizarse, lo que complica la solución.

Otra de las formas propuestas es recoger todas las cookies que el usuario tiene almacenadas en el navegador y enviarlas junto con la petición. El método utilizado para enviarlas es mediante un fichero con el comando WGET [18]. El formato de fichero utilizado es el mismo que utiliza Netscape para manejar las cookies del usuario y que es compatible con el comando con el que se realiza la petición las páginas, WGET [18]. El problema de esta propuesta es cómo recoger estas cookies. Por JAVASCRIPT [20] se hace imposible esta tarea, puesto que solo permite acceder a las cookies que han sido creadas por el mismo host que el que hace la petición, y puesto que el objetivo es

validar páginas ajenas al servidor de instalación de la aplicación, esta solución no es válida. Otra forma de obtener las cookies es accediendo directamente al fichero de cookies del navegador que se esté utilizando, aunque esto haría que la aplicación fuese totalmente dependiente del navegador y de su propio formato del fichero utilizado para almacenar estas cookies. Por último, la solución sería recuperar las cookies a través de una extensión de Mozilla Firefox y crear con esto el fichero que se enviará junto con el comando WGET [18] aunque limitará el uso de la funcionalidad de las cookies a sólo los casos en los que la entrada al sistema sea a través de la extensión creada para Mozilla Firefox.

4.4.5. Modificación de las páginas

En esta parte del sistema, el único problema que supone un punto de discusión de posibles soluciones es si utilizar un módulo totalmente ajeno a WAEX [1] o no. Si se realiza como una parte de WAEX [1] se garantiza estabilidad y utilizar un mismo diseño para todo el proyecto. La segunda solución es realizar un módulo, de forma que lo hace totalmente independiente e incluso con la posibilidad de desarrollar en lenguajes distintos. Esto permitiría al sistema paralelizar las funciones de validación y aplicación de los cambios en las validaciones anteriores. Es decir, con este módulo a parte, se podría hacer la llamada a los cambios, y mientras estos se llevan a cabo, el usuario podría seguir haciendo peticiones de validación del resto de las páginas. Incluso, este módulo podría estar ejecutándose en otro servidor para garantizar los tiempos de espera menores. Además, si en un futuro este módulo supone un tiempo de espera muy alto, algo que podría ocurrir, y así se permitiría ampliar el conjunto de reglas que pueden ser modificadas online, sin realizar cambios en WAEX [1].

4.4.6. Salida del sistema

La salida del sistema solo supone la decisión del formato de la interfaz que será detallado posteriormente mediante prototipos.

4.5. Selección de la solución a desarrollar

Finalmente se detallan las conclusiones acerca de qué se quiere desarrollar y su solución. Como en el punto anterior, todas las decisiones son en base a los distintos módulos en los que se ha dividido el proyecto.

4.5.1. Recorrido de páginas

El recorrido de las páginas se realizará utilizando el comando WGET [18] ya que tiene las mismas opciones que el comando GET [17] y ayuda a solucionar uno de los problemas que se requieren resolver, el soporte a cookies. Con el comando elegido es posible enviar peticiones acompañadas de un fichero con las cookies oportunas. El manual de ambos comandos se puede encontrar entre las referencias al final de este documento.

4.5.2. Validación de WAEX y validación de color

Para garantizar la estabilidad del sistema y que esté actualizado según la última versión de WAEX [1] se va a adoptar la solución de enviar las peticiones directamente por http y analizar el resultado de la página que devuelve como respuesta. Por el mismo motivo, la validación de color será un petición vía post al validador <http://www.accesskeys.org/tools/color-contrast.html>.

En ambos casos, en las salidas se puede separar la información relevante (en el centro de la página) del resto. En la Figura 6 y en la Figura 7 se muestran los ejemplos de las salidas de ambos validadores sobre una página de prueba creada para este caso. En el caso de la Figura 6 será importante todo lo que se muestra, y para la Figura 7 únicamente interesa conocer lo que hay dentro del recuadro rojo.

Accessibility report for <http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html>

Report summary		
Automated tests	Result	Level
WCAG Checkpoints priority 1	FAILED	
WCAG Checkpoints priority 2	FAILED	
WCAG Checkpoints priority 3	FAILED	
Checkpoints not in WCAG	FAILED	

WCAG Checkpoints priority 1

Areas with no alt attribute

- Line 13. <area shape="polygon" coords="19,44,45,11,87,37,82,76,49,98" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" />
- Line 14. <area shape="rect" coords="128,132,241,179" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" />
- Line 15. <area shape="circle" coords="68,211,35" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" />

WCAG Checkpoints priority 2

Presentational attributes translatable to CSS

- Line 12. uses HTML markup (not deprecated yet) for presentation. Use CSS instead.

Deprecated elements or presentation elements

- Markup must be valid

WCAG Checkpoints priority 3

Areas having no redundant link

- Line 13. <area shape="polygon" coords="19,44,45,11,87,37,82,76,49,98" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" /> requires a redundant link to <http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html> somewhere else in the document.
- Line 14. <area shape="rect" coords="128,132,241,179" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" /> requires a redundant link to <http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html> somewhere else in the document.
- Line 15. <area shape="circle" coords="68,211,35" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" /> requires a redundant link to <http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html> somewhere else in the document.

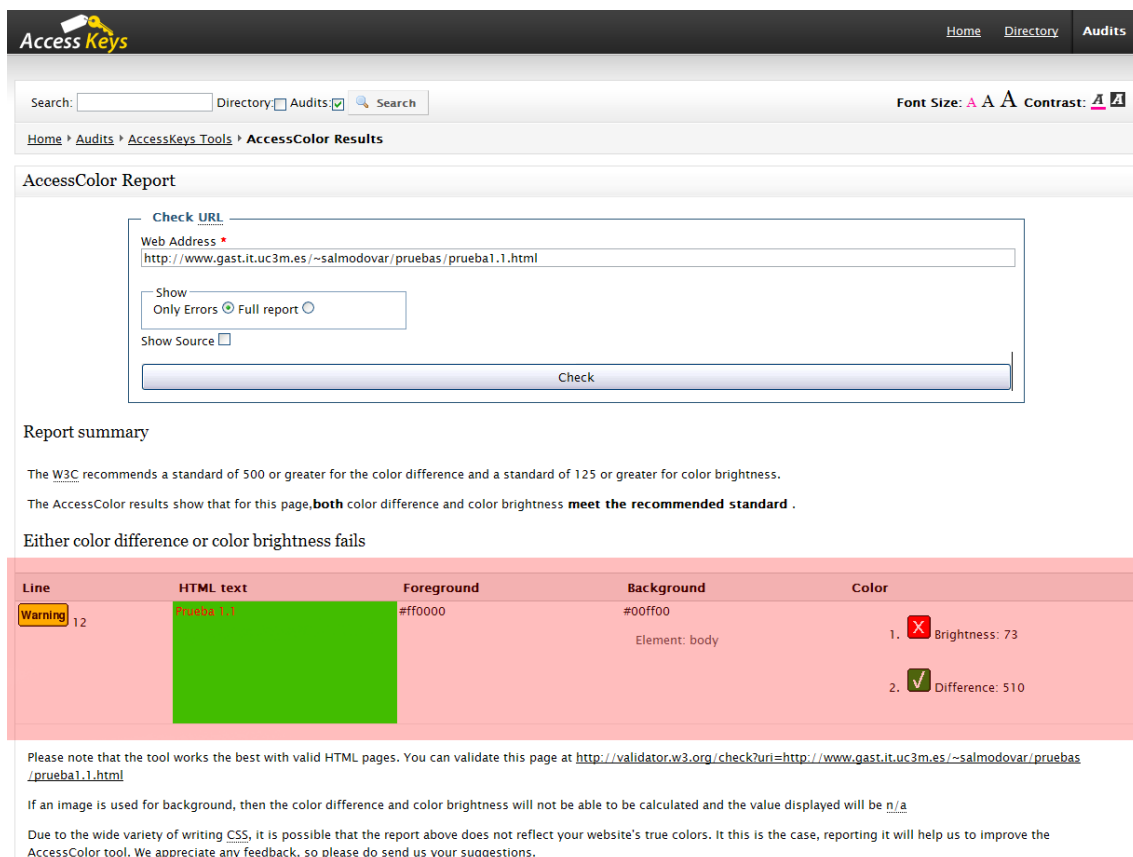
Checkpoints not in WCAG

Maps forbidden

- Line 12.
- Line 13. <area shape="polygon" coords="19,44,45,11,87,37,82,76,49,98" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" />
- Line 14. <area shape="rect" coords="128,132,241,179" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" />
- Line 15. <area shape="circle" coords="68,211,35" href="http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html" />

Figura 6 Salida de la actual aplicación WAEX

En el validador WAEX [1], la salida es completamente reutilizable, y el formato de ésta se mantendrá para no cambiar el valor del estilo en la información que devuelve la ruta de página.



The screenshot shows the AccessKeys website interface. At the top, there's a navigation bar with 'Home', 'Directory', and 'Audits' links. Below it is a search bar and a font size/contrast selector. The main content area is titled 'AccessColor Report'. It contains a form to input a URL and select report options. The report summary states that the W3C recommends a standard of 500 or greater for color difference and 125 or greater for color brightness. The AccessColor results show that for this page, both color difference and color brightness meet the recommended standard. However, a warning is displayed: 'Either color difference or color brightness fails'. This is followed by a table of results.

Line	HTML text	Foreground	Background	Color
Warning 12	Prueba 1.1	#ff0000	#00ff00 Element: body	1. Brightness: 73 2. Difference: 510

Please note that the tool works the best with valid HTML pages. You can validate this page at <http://validator.w3.org/check?uri=http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html>

If an image is used for background, then the color difference and color brightness will not be able to be calculated and the value displayed will be n/a

Due to the wide variety of writing CSS, it is possible that the report above does not reflect your website's true colors. If this is the case, reporting it will help us to improve the AccessColor tool. We appreciate any feedback, so please do [send us](#) your suggestions.

Figura 7 Salida de la actual aplicación de validación de color Access Keys

En cuanto al validador de color, sólo interesa la tabla de resultados de errores, que ha sido señalada en el cuadro rojo.

4.5.3. Soporte a cookies

El soporte a cookies solo estará disponible en los casos en los que la entrada al sistema sea a partir de la extensión de Mozilla Firefox. Esto permitirá recoger las cookies que hay almacenadas en el sistema y ser enviadas por peticiones con el comando anterior.

Si la entrada no es a partir de la extensión que se creará para Mozilla Firefox, el sistema no tendrá la opción de soporte a cookies.

4.5.4. Modificación de las páginas

La modificación de las páginas se realizará a través de una librería externa con el fin de paralelizar el proceso en un futuro. Esta nueva librería formará parte del proyecto y el detalle de su diseño se explicará en el capítulo siguiente.

En la siguiente imagen (Figura 8) se muestra un esquema de la relación y comunicación de los módulos que forman el sistema:

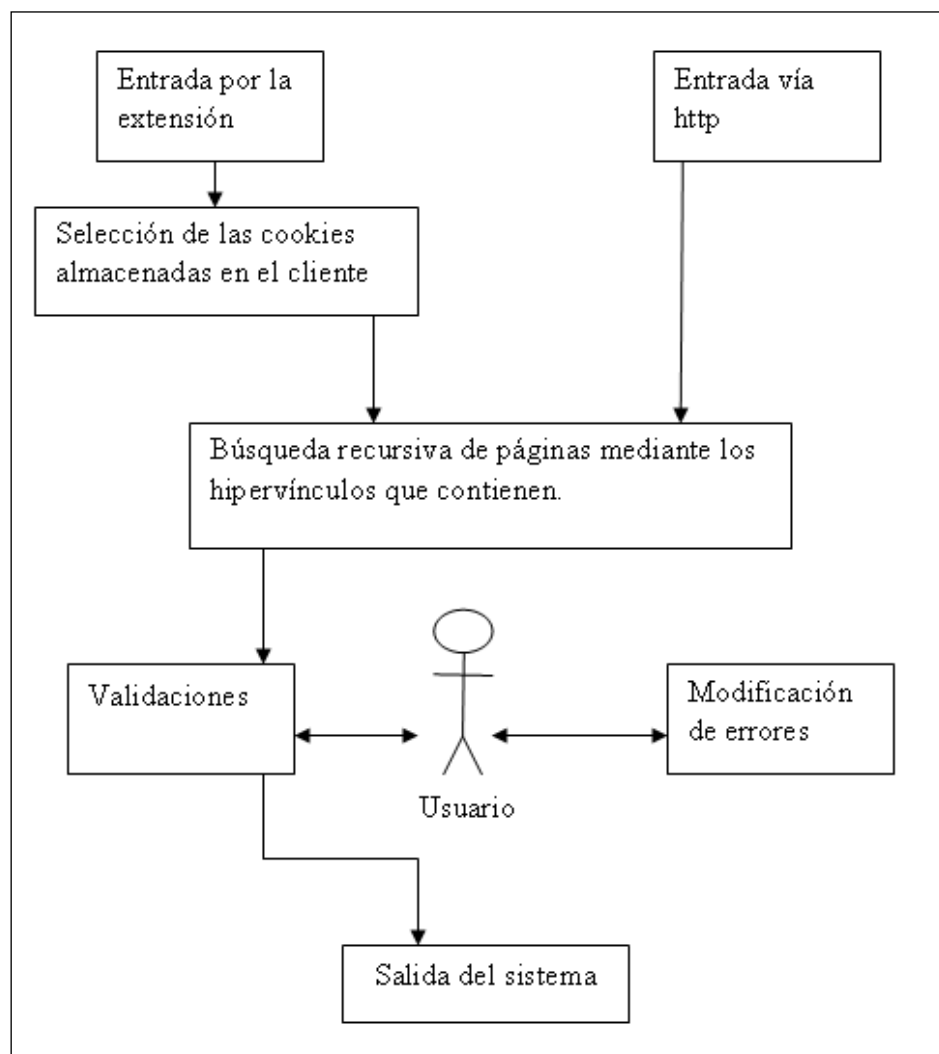


Figura 8 Esquema de funcionalidad de WAEX2

5. CAPÍTULO IV: DESARROLLO

5.1. Escenario

La nueva aplicación tendrá diferentes escenarios, para separar la parte previa a la validación de todas las funcionalidades restantes. A continuación se detallan las dos interfaces que serán utilizadas en el sistema. Ambas serán presentadas como prototipos del resultado final.

5.1.1. Interfaz de entrada al sistema

La interfaz de entrada al sistema será aquella que el usuario verá en primer lugar. Sin este paso, no existirá ningún tipo de validación.


Lo más importante de este punto, es que el resultado sea lo más similar posible a WAEX [1]. Además será una interfaz muy simple y será importante tener en cuenta las entradas que deben ser exactamente las mismas que WAEX [1].

El esquema de ésta será el siguiente:

CABECERA CON EL TITULO DE WAEX
Entrada de datos: URL
Entrada de datos: Max. Alt. Longitud máxima permitida en los atributos alt para elementos sin atributo longdesc
Entrada de datos: Max. Title. Longitud máxima permitida en los atributos title para elementos sin atributo longdesc
Entrada de datos: Max. Th. Longitud máxima permitida para las cabeceras de las tablas sin un abbr
Entrada de datos: Max. Option. Máximo número de opciones en grupos de opciones.
Entrada de datos: Max. Fields. Máximo número de campos agrupados en campos de un formulario.

[Tabla 34 Descripción de los campos de entrada a WAEX](#)

Esta es una estructura aproximada. En el caso de la extensión de Mozilla Firefox se ha conseguido con la misma estructura, tal y como se muestra en la Figura 9.



WAEX 2
Web Accessibility Evaluator in a single
XSLT file

URL:

Max. alt

Max. title

Max. th

Max. option

Max. fields

Run WAEX

Figura 9 Prototipo de entrada a WAEX2 a través de la extensión.

En cambio para el caso de entrada por Web, se ha mantenido la misma entrada que utiliza WAEX [1] para validar con su propio xslt, tal y como se puede ver en la Figura 10.

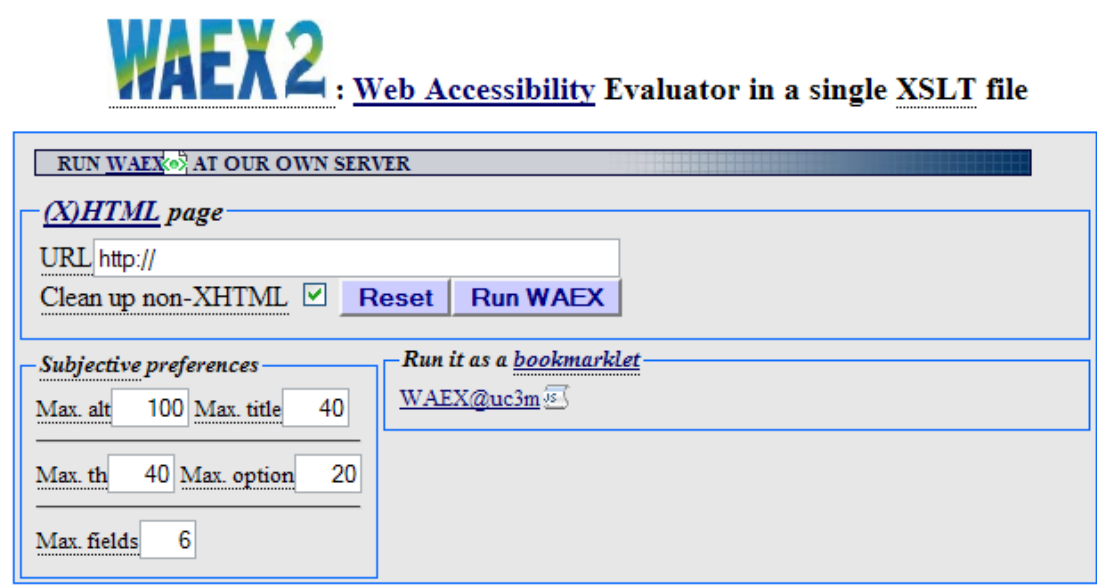


Figura 10 Prototipo de entrada al sistema WAEX2 vía http

5.1.2. Interfaz principal de validación

Las características de esta interfaz, deben ser similares a las propuestas en el requisito de interfaz RSI-07. La interfaz ha sido diseñada a través del prototipo que muestra la Figura 11:

Tipo de validación y página validada		Funcionalidades:
<div> <div>Enlace para cambiar tipo de validación</div> <div>Aplicar cambios</div> </div> <div>Resultado de la validación</div>		Validar página siguiente en la lista. Validar página anterior en la lista. Lista completa para seleccionar la validación de una de ellas. Descargar de la solución. Restaurar página actual.
		Detalles de la validación

Figura 11 Prototipo de la interfaz de validación de WAEX2

A este diseño se le añade una cabecera en la parte superior donde irá el título de la aplicación, WAEX 2. En “Tipo de validación y página validada” se dará información de

la URL de la página cuyo resultado se muestra en la parte inferior y si ha sido validada por WAEX [1] (indicado con la palabra código) o ha sido una validación referente al color. Se podrá cambiar el tipo de validación mediante un enlace que aparecerá en la parte superior de Resultado de validación.

En la parte central aparecerá el resultado de la validación seleccionada para la página que indica la cabecera. Aparece también un enlace para aplicar los cambios que se han realizado en su post-validación, y guardarlos hasta volver a visitar esta página o descargar la solución. El motivo de que esta funcionalidad se encuentre separada del resto es debido a que se considera la más importante entre las que se muestran en el prototipo, y se relaciona con el resultado de la validación, que dará opciones sencillas para modificar las páginas.

Por último se reserva un espacio para la documentación del validador que está funcionando en el momento, ya sea WAEX [1] o el utilizado para analizar el color.

El resultado será similar a la Figura 12:

Color report summary for page
<http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba.html>

WAEX Color

Summary color report

The W3C recommends a standard of 500 or greater for the color difference and a standard of 125 or greater for color brightness.

The AccessColor results show that for this page, both color difference and color brightness doesn't meet the recommended standard.

Found errors

Line	HTML text	Foreground	Background	Color
Fail 12	prueba 1.1	blue Element: a Change Color	#00ff00 Element: body Change Color	✓ Brightness : 141 ✗ Contrast : 252
Fail 13	prueba 4.3	blue Element: a Change Color	#00ff00 Element: body Change Color	✓ Brightness : 141 ✗ Contrast : 252
Fail 14	prueba 5.5	blue Element: a Change Color	#00ff00 Element: body Change Color	✓ Brightness : 141 ✗ Contrast : 252
Fail 15	prueba 13.1	blue Element: a Change Color	#00ff00 Element: body Change Color	✓ Brightness : 141 ✗ Contrast : 252
Fail 16	prueba font	blue Element: a Change Color	#00ff00 Element: body Change Color	✓ Brightness : 141 ✗ Contrast : 252

Download solution
Discard changes

Select a page
<http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba.html>
<http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba1.1.html>
<http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba4.3.html>
<http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba5.5.html>
<http://www.gast.it.uc3m.es/~salmodovar/pruebas/prueba13.1.htm>

Previous page Next page

The calculation algorithm used to decide if the use of a pair of color is correct is the algorithm stipulated by W3C and it can be consulted at the following link: [Algorithm for the brightness and contrast color calculation](#). The color validation search for differences between background and foreground colors. This tool is not able to evaluate the text over images or flash elements. If an image is use as background, the color and brightness difference will not be calculated and the value which is shown in the summary report will be n/a.

Figura 12 Ejemplo de interfaz de validación

5.2. Selección de las herramientas software

La selección de las herramientas software es la tarea consistente en elegir qué lenguajes de programación y herramientas serán necesarios para la implementación del sistema. La principal característica que se debe tener en cuenta es elegir el lenguaje más apropiado.

Esta decisión debe tomarse según la naturaleza del programa, su propósito y los requisitos. Tal como se ha descrito en los requisitos, debe ser una aplicación Web por lo que distinguimos entre lenguajes que se ejecutarán en el cliente y los que se ejecutarán en el servidor.

De cara al cliente, todas las páginas serán desarrolladas en HTML [7]. En estas páginas se definirá el contenido de la información. El estilo de la página será incorporado a cada documento HTML [7] mediante un enlace a una hoja de estilo externa CSS [9].

El dinamismo en la parte del cliente se resolverá mediante llamadas a objetos JAVASCRIPT [20].

Se ha prestado la máxima atención en devolver páginas con la máxima accesibilidad posible. Sin embargo, al ser un sistema que necesita de interacción con el usuario, ha sido imposible garantizar al 100% la accesibilidad Web debido a la necesidad de utilizar JAVASCRIPT [20] para este cometido.

Para el desarrollo de la extensión de Mozilla Firefox, la única vía posible es desarrollarlo en un lenguaje basado en XML y creado especialmente para generar nuevas extensiones y que puedan ser añadidas a cualquier navegador Mozilla Firefox según está documentado en . Este lenguaje es XUL [21], cuyo significado es XML based User interface Language.

Para la ejecución del sistema en el lado del servidor se utilizará PHP [22], cuya respuesta será el documento HTML [7] que llegará al cliente. El código PHP [22] se ejecutará en el servidor. Se ha decidido utilizar para este proyecto este lenguaje por diferentes motivos, entre los que se destacan como más importantes:

- Permite dinamismo y es fácil realizar peticiones.
- El uso de PHP [22] implica que ésta sea una aplicación Web, y por tanto, hace que ésta sea una aplicación multiplataforma.
- Es *Open Source*.
- PHP [22] permite ejecutar comandos del sistema mediante una función predefinida. De este modo, se puede ejecutar los comandos de consola GET [17] y WGET [18].

El servidor donde estará instalada la aplicación y será utilizado para el desarrollo, será el facilitado por el departamento de Telemática para el desarrollo del proyecto y cuyo host es: www.gast.it.uc3m.es. La cuenta está abierta bajo un servidor Linux en el cual se ha instalado PHP [22] en la versión 4.4.4, aunque no es la más actual.

Por último, quedan por definir las herramientas utilizadas para implementar la librería de cambios. Ya que se ha decidido que esta librería sea un módulo independiente, facilita su integración en el proyecto a pesar de decidir implementarla en un lenguaje que no fuese PHP [22]. Se ha decidido implementarla en c++ [23] por varias razones. La primera de ellas es que c++ [23] es un lenguaje muy parecido en sintaxis a PHP, y ya que éste permite ejecución de programas por línea de comandos en el servidor, su integración en el sistema era bastante sencilla. Además, como se ha considerado que para la posterioridad el trabajo de esta librería pueda verse incrementada y se puedan añadir fácilmente nuevas mejoras en la ayuda a la post-validación, realizarlo en c++ [23] y por separado permitirá una ejecución mucho más rápida y con la opción de que sea un proceso paralelo a las respuestas y trabajos que se continúen realizando entre cliente-servidor con la validación de otras páginas. De esta forma, el tiempo de respuesta del usuario se verá disminuido. Por último, la ventaja de este lenguaje es que existen librerías de análisis XML muy apropiadas para el trabajo que debe realizar, con

el que será muy rápido analizar y modificar los nodos de los que se componen las páginas HTML [7].

En las referencias del documento puede consultarse la información utilizada para el desarrollo con estas herramientas.

5.3. Arquitectura de la solución

La arquitectura define los módulos, comunicaciones y estructuras de cada uno de los módulos y componentes que existirán en el sistema.

Se presenta en la Figura 13 un primer esquema de cada uno de los módulos y cómo van a interactuar, y a continuación serán descritos cada uno de ellos.

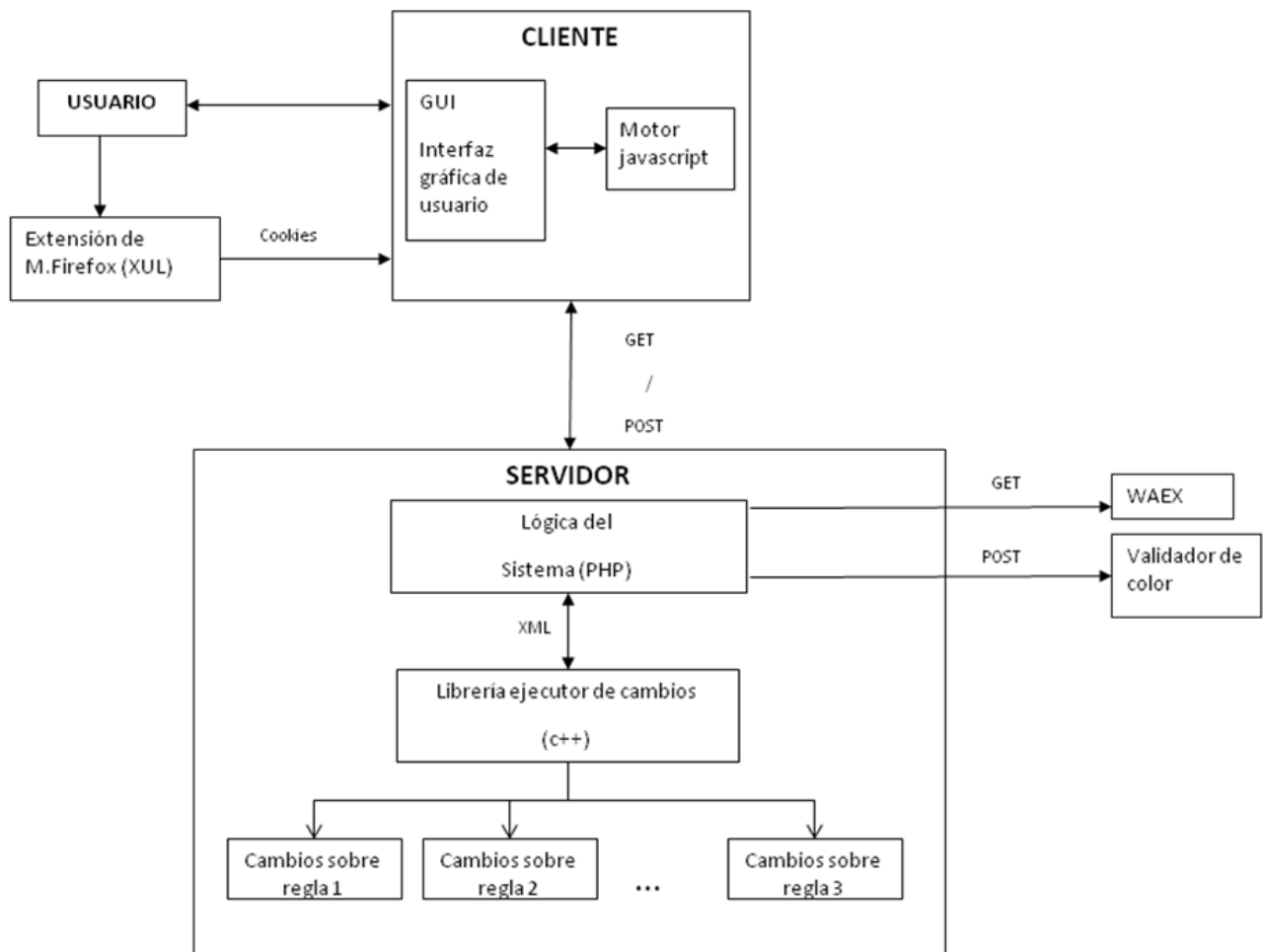


Figura 13 Estructura de los diferentes módulos de la aplicación.

El usuario realiza la entrada al sistema por dos posibles vías: a través de la extensión de Mozilla Firefox o bien por una página suministrada a la interfaz gráfica por parte del servidor.

Una vez el usuario entra en la extensión de Mozilla Firefox y envía los datos, (éste sólo envía los datos de la petición por lo que la comunicación es unidireccional), la extensión se encarga de generar todas las variables del entorno de entrada al sistema, comunicándose con el navegador a través de cookies. Las cookies son recogidas antes de llegar al cliente para que éste las envíe y guarde al servidor. Otra de las variables de entorno que crea esta extensión es la destinada a dar el nombre de la carpeta donde se guardarán todos los datos de validación del usuario. Cada vez que un usuario entra en la aplicación, se crea una nueva carpeta para éste con un número identificativo.

Desde la segunda entrada posible al sistema se genera la misma situación anterior excepto que no se dispone de un fichero de cookies y por lo tanto no es enviado en las peticiones durante el proceso de validación.

En el cliente se muestran al usuario todas las interfaces de usuario que genera el sistema. Todas ellas son mostradas mediante HTML [7]. El usuario interactúa y manda las peticiones a través de javascript [20]. Todas las peticiones desde el cliente al servidor son enviadas mediante formularios por vía GET y POST.

La lógica del sistema es el módulo más importante de la arquitectura. En éste se generan todas las páginas que son enviadas al cliente. Éstas estarán en base a dos puntos:

- Un proceso de búsqueda de todas las páginas a las que se llega mediante navegación de hipervínculos.
- Es el encargado de realizar todas las peticiones al exterior, tanto para recuperar validaciones contra WAEX [1], contra el validador de color o bien para enviar la solicitud de cambios, cuando hayan sido solicitados por el cliente.

Las respuestas de ambos validadores son analizadas y modificadas en este módulo para adaptarlas a la salida que se espera en la nueva aplicación, quitando cabeceras, información no necesaria y permitir entradas al sistema para cubrir los requisitos de ayuda a la postvalidación. Esta ayuda mediante formularios, enviará los datos que ha introducido el usuario del cliente al servidor, y de nuevo la lógica del sistema se encarga

de recuperarlos. Ésta transforma todos los datos de cambios recogidos en un fichero xml que será enviado como entrada a la biblioteca que realiza los cambios.

La biblioteca de cambios divide la estructura de todas las páginas, separando el estilo de la estructura de modo que se tienen dos tipos de árboles compuestos por nodos XML, uno para el CSS [9] y otro para los nodos sin estilo de HTML [7]. Realizará los cambios a partir de la obtención de todos los nodos del árbol que forman el código HTML [7] y buscando las cadenas e identificadores de los nodos para los que se solicita algún cambio. Estos nodos fueron previamente tratados en la lógica del sistema, para incorporar un atributo que los identificase e hiciese sencilla su recuperación y reemplazo.

La biblioteca de cambios puede ser ejecutada desde fuera de la aplicación, siempre y cuando se le pasen las entradas correctas como se indica en la siguiente sintaxis de ejecución:

wcag *html_path css_path changes_xml*

html_path es la ruta del fichero que contiene el código HTML [7] de la página sobre la que se quiere realizar cambios.

css_path es la ruta del fichero que contiene una representación en nodos XML del CSS [9] de la página.

changes_xml es el fichero XML que contiene los cambios que deben ser realizados en el código del fichero HTML [7] del primer elemento.

Los cambios deben ser un fichero XML cuya estructura se corresponde con la siguiente definición de tipo de documento (DTD) que se muestran en la Tabla 35:

```
<!ELEMENT cambios (img, idioma, tables, text, frame)>
<!ELEMENT img (nodo_img*)>
<!ELEMENT nodo_img (imgAlt, textAlt)>
<!ELEMENT imgAlt (#PCDATA)>
<!ELEMENT textAlt (#PCDATA)>

<!ELEMENT idioma (nodo_idioma*)>
<!ELEMENT nodo_idioma (idiomaSel)>
<!ELEMENT idiomaSel (#PCDATA)>

<!ELEMENT tables (nodo_tablas*)>
<!ELEMENT nodo_tablas (sumAlt, textAlt)>
<!ELEMENT sumAlt (#PCDATA)>

<!ELEMENT frame (nodo_frame*)>
<!ELEMENT nodo_frame (frameTitle, textFrame)>
<!ELEMENT frameTitle (#PCDATA)>
<!ELEMENT textFrame (#PCDATA)>
```

Tabla 35 DTD del fichero de cambios

En el siguiente punto del documento, se detallan todos los módulos de un modo más específico y con vistas a la implementación.

5.4. Diseño e implementación de las bibliotecas

Para facilitar la explicación del diseño e implementación de las bibliotecas, se hará una explicación por cada uno de los módulos detallando las secuencias de ejecución, clases y funciones principales que los forman.

5.4.1. Lógica del sistema

Las funciones principales de este sistema se ejecutan de un modo secuencial. El primer paso es realizar el recorrido de las páginas y obtener todas las URL's a las que se pueden llegar a través sus hipervínculos, siempre y cuando éstas pertenezcan al mismo dominio que la página inicial proporcionada por el usuario.

El recorrido de páginas es un proceso recursivo. Por cada página se realizan varios pasos, tal como se explica en el siguiente gráfico (Figura 14):

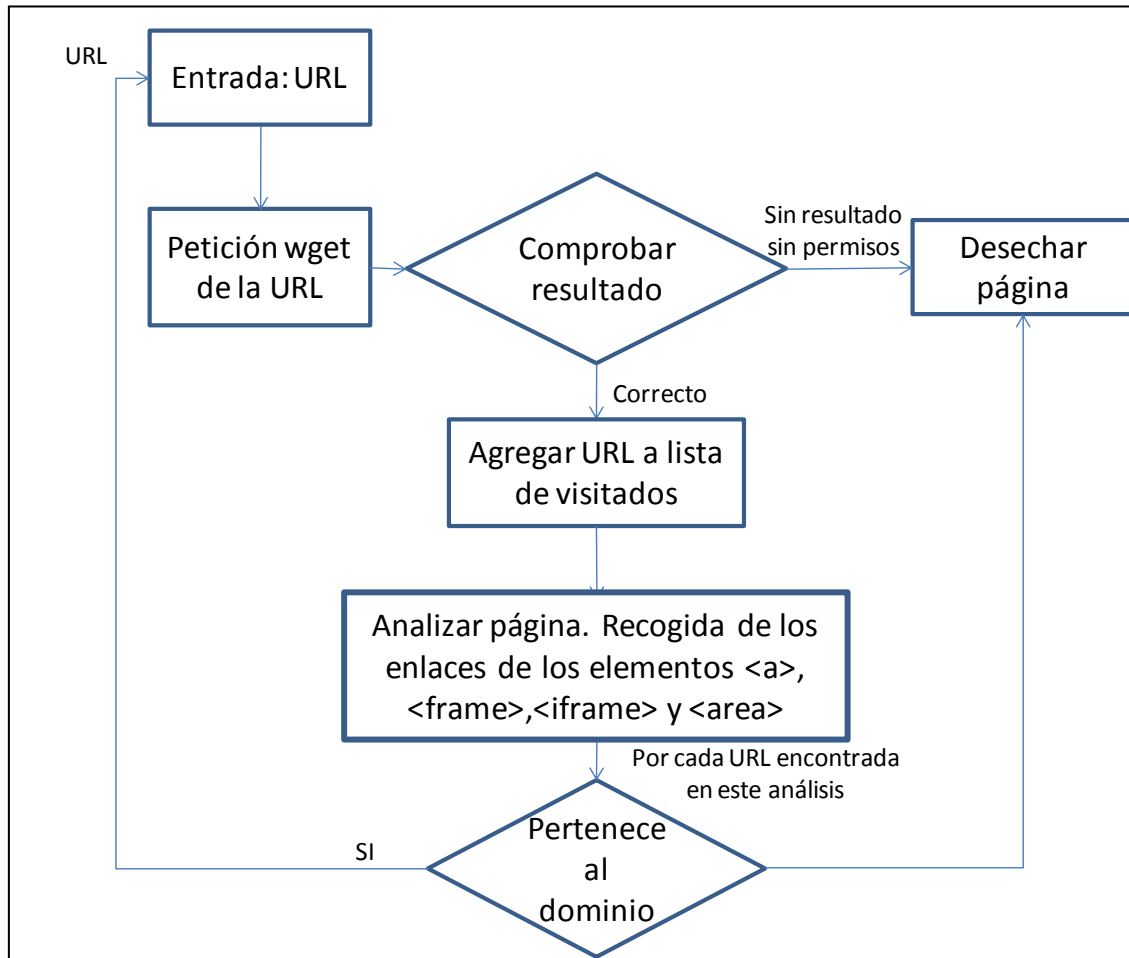


Figura 14 Esquema de funcionamiento para recopilar los enlaces del dominio

Una página no será recogida en el listado si no devuelve una página HTML [7] o si no existen permisos para acceder a ella.

Se considerará que una página pertenece al dominio siempre y cuando el nombre del servidor coincida con el de la página inicial. Así, se garantiza que existirá un fin en el recorrido de los enlaces.

Una vez terminado este proceso, es decir, cuando no se encuentran nuevos enlaces, o los enlaces encontrados están en la lista de visitados se procede al siguiente paso. Para esto, todas las URL que se han encontrado se envían al servidor y éste las almacena en un fichero de texto, una por línea. El nombre de este fichero es “listadoPaginas” y se almacena en la carpeta creada para este usuario, cuyo nombre es un número aleatorio generado al inicio de todo el proceso de validación.

A partir de este momento, el proceso de validación ha creado unas variables que serán enviadas mediante POST en cada momento, y serán las responsables de decidir qué debe realizar y validar el sistema en cada momento. Las variables son las siguientes (Figura 15):

Variable	Descripción
Dir	Indica el nombre del directorio utilizado para guardar todos los datos de la validación
Numero de página	Indica el índice de la página que se pide validar, y cuyo orden corresponderá con la línea del fichero listadoPaginas, donde se almacena la URL de ésta página.
Tipo de validación	Se distinguen dos tipos de validación, la que realiza WAEX (código) o la de color. Esta variable indicará si se desea realizar una u otra sobre la página pedida.

Figura 15 Descripción de las variables generales del sistema

Todas las páginas de las cuales se pide petición de validación, primero son obtenidas mediante el comando WGET [18] y almacenadas en el propio servidor. Con esto se consigue que las peticiones de cookies se realicen sólo una vez y se trabaje sobre la copia (con posibles cambios) del servidor. Además, de este modo no tiene que realizar continuamente la petición de la misma página puesto que ya está en local, y el proceso de validación es más rápido. Se almacenan en el directorio que indica la variable Dir. Para evitar que existan dos páginas con URL's distintas pero con el mismo nombre, las páginas son almacenadas en la misma estructura de directorios que indica la URL quitando el dominio. De esta forma se ayuda a identificar qué páginas se han modificado cuando se descargue el usuario la solución.

Una vez realizada la petición a uno de los distintos validadores, se procede al análisis de la respuesta. Su cometido será seleccionar únicamente la información que interesa de la respuesta que devuelve WAEX [1] o el validador de color y montar la página que será enviada al cliente. Además de seleccionar la información relevante, es modificada y tratada para permitir los cambios por parte del usuario.

Los cambios en la validación de color se realizan seleccionando el enlace *Cambio en el Color* de texto o de fondo de cada uno de los elementos en los que ha dado error la validación. El color es elegido por parte del usuario mediante un selector de color como puede verse en la Figura 16.

Line	HTML text
Fail 12	prueba 1.1
Fail 13	prueba 4.3
Fail 14	prueba 5.5
Fail 15	prueba 13.1
Fail 16	prueba font

Figura 16 Ejemplo de la funcionalidad para cambiar color.

Cada uno de los enlaces “Cambiar Color” lleva asociada la siguiente información (oculta al usuario):

- Línea del código de la página origen donde se encuentra el error.
- Texto donde se encuentra el error. En la imagen (Figura 16), se corresponde con el texto de la columna “HTML text”.
- Un identificador para poder situarnos en los datos del error en futuras operaciones.

De esta forma, todos los cambios que se hayan realizado, se envían al servidor y éste los trata para enviarlos mediante XML a la biblioteca de cambios.

Ocurre algo parecido en el caso de la validación de WAEX [1]. Por cada error que se considera posible de modificación tras la validación, se asocia a una entrada de datos y mantiene información de la parte de código a la que se referencia dentro del documento. Estos posibles cambios y su formato de entrada de datos que se incorporan son los siguientes:

WCAG 1.1: Falta atributo alt

Este error ocurre cuando aparece un elemento gráfico dentro del código HTML [7] que no lleva asociado un atributo *alt* con la descripción textual de dicho elemento. Incorporar este atributo permite conocer el contenido del elemento en caso de no poder ser visualizado.

Solucionar este error consiste únicamente en dar esta descripción que será incorporada al nodo HTML [7]. Para la solución gráfica, se añade una entrada de texto para que el usuario pueda escribir la descripción del elemento. En la imagen siguiente (Figura 17) se muestra un ejemplo de cómo quedaría este tipo de error:



Figura 17 Ejemplo de salida adaptada para cambiar errores de la regla WCAG1.1

WCAG 1.2: Falta atributo alt

Este error ocurre cuando un mapa de imágenes dentro del código *HTML* [7] no lleva asociado un atributo *alt* con la descripción textual de este elemento. Incorporar este atributo permite conocer el contenido de las imágenes en caso de no poder ser visualizado o que los usuarios tengan algún tipo de discapacidad visual.

Solucionar este error consiste únicamente en proporcionar una descripción textual que será incorporada al código de la página en forma de atributo *alt* para el elemento. En la imagen anterior (Figura 17), el primer error que se muestra se correspondería con uno producido por esta regla. Como se puede observar, en este caso, también se pide de una descripción para la imagen que representa.

WCAG 2.2 [10]: Contraste y brillo:

Esta regla no se muestra como error en WAEX [1]. La validación de esta regla se ha explicado anteriormente, e irá de forma paralela a la de WAEX [1] pero nunca juntas. Los cambios que se pidan ejecutar se realizarán a través de la librería de cambios indicando el contenido anterior, color de fondo y de fuente nuevos.

WCAG 4.3: Falta idioma del documento

Este error aparece cuando una página *HTML* [7] no tiene información del idioma en la que está escrita.

Se incorpora una entrada de datos para que el usuario seleccione de una lista el idioma correspondiente al documento. Estas opciones se pueden editar, ya que son almacenados en el fichero idiomas del sistema, uno por cada línea. Este fichero se encontrará en la carpeta donde esté instalado WAEX2, con el nombre idiomas.txt.

En la imagen siguiente (Figura 18) se muestra un ejemplo del desplegable que se mostraría al usuario en caso de tener este error.

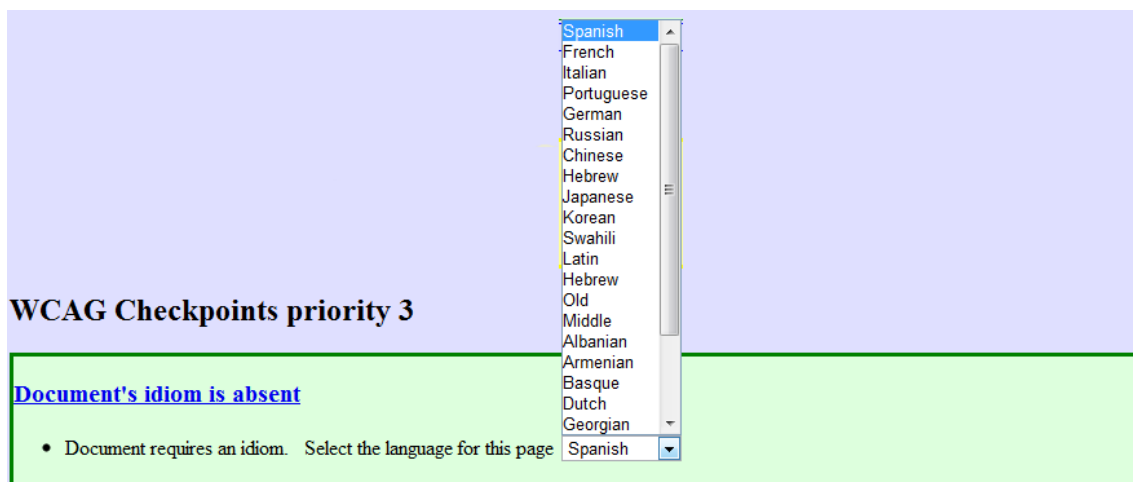


Figura 18 Ejemplo de salida adaptada para cambiar errores de la regla WCAG4.3

WCAG 5.5: Falta atributo summary para la tabla

Este error aparece al encontrar entre el código un elemento *table* que no tiene el atributo *summary*, con el texto que describe el contenido de la tabla.

Se incorpora una entrada de datos para que el usuario escriba la descripción que irá en el atributo *summary* de la tabla por la que se ha producido el error. En la siguiente imagen (Figura 19) se muestra un ejemplo de este error con el formato de su posible solución:

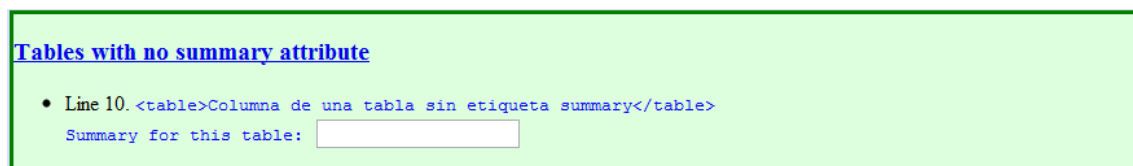


Figura 19 Ejemplo de salida adaptada para cambiar errores de la regla WCAG5.5

WCAG 12.1: Falta atributo title

Este error está presente en la validación cuando se encuentra un elemento de tipo FRAME que no contiene en su definición el atributo title o éste está vacío. Este atributo será importante en cuanto a que sin él sería imposible identificar los distintos marcos de los que se compone una página. Como aclaración, aunque se pretenda identificar este error, se debería evitar el uso de elementos FRAME ya que no son accesibles.


Este error se solucionaría con la entrada del texto que contiene el título del marco y que será añadido como atributo al texto.

WCAG 13.1: Falta elemento o información

Este error aparece cuando un elemento importante dentro de la estructura HTML no está presente en el documento, por ejemplo el elemento *title* (aunque no es el único).

Son elementos que se consideran imprescindibles y, o bien no aparecen, o no tienen datos en su contenido.

Se incorpora una entrada de datos para cada uno de ellos para que el usuario pueda escribir la información requerida, similar a la Figura 20.



Deprecated elements or presentation elements

- Markup must be valid

Elements requiring a pronounceable text

- Line 7. <title />
Alternative text for this element:

Figura 20 Ejemplo de salida adaptada para cambiar errores de la regla WCAG13.1

Font elements

Se considera que los elementos *font* como obsoletos, y producen un error de validación. Este error es mostrado al usuario con el texto: “*Deprecated elements or presentation elements. Markup must be valid*” aunque no es la única razón por la que puede aparecer este mensaje. Existirá una opción para que el usuario decida si quiere realizar cambios sobre estos elementos, que consistirá en cambiar estos elementos por etiquetas *span* y asociarles a un nuevo estilo CSS [9] con todas las características del tipo de fuente que tenía el elemento anterior. La opción de mejora se muestra en un cuadro como el de la Figura 21.

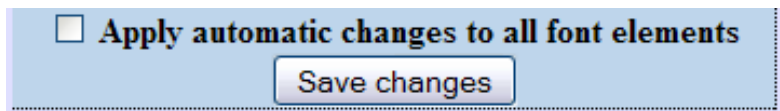


Figura 21 Opción para solucionar problemas de uso del elemento obsoleto FONT

5.4.2. Librería ejecutor de cambios

Cada uno de los elementos anteriores es tratado y modificado para utilizar los nuevos datos y dar solución a los problemas generados en la validación. Los elementos que se muestran, un identificador, el tipo de cambio que se va a aplicar al elemento, el texto completo que constituye al elemento y el valor introducido por el usuario para dicho cambio. Esta librería se encargará de realizar los cambios a dichos elementos. Su entrada será un fichero XML donde se engloban todos los datos anteriores, con un nodo por cada cambio que se vaya a realizar. Este XML es enviado a la librería que modificará la página siguiendo cada uno de los elementos de los que se forma este fichero de cambios.

La librería sigue una estructura de clases según se muestra en la siguiente imagen (Figura 22).

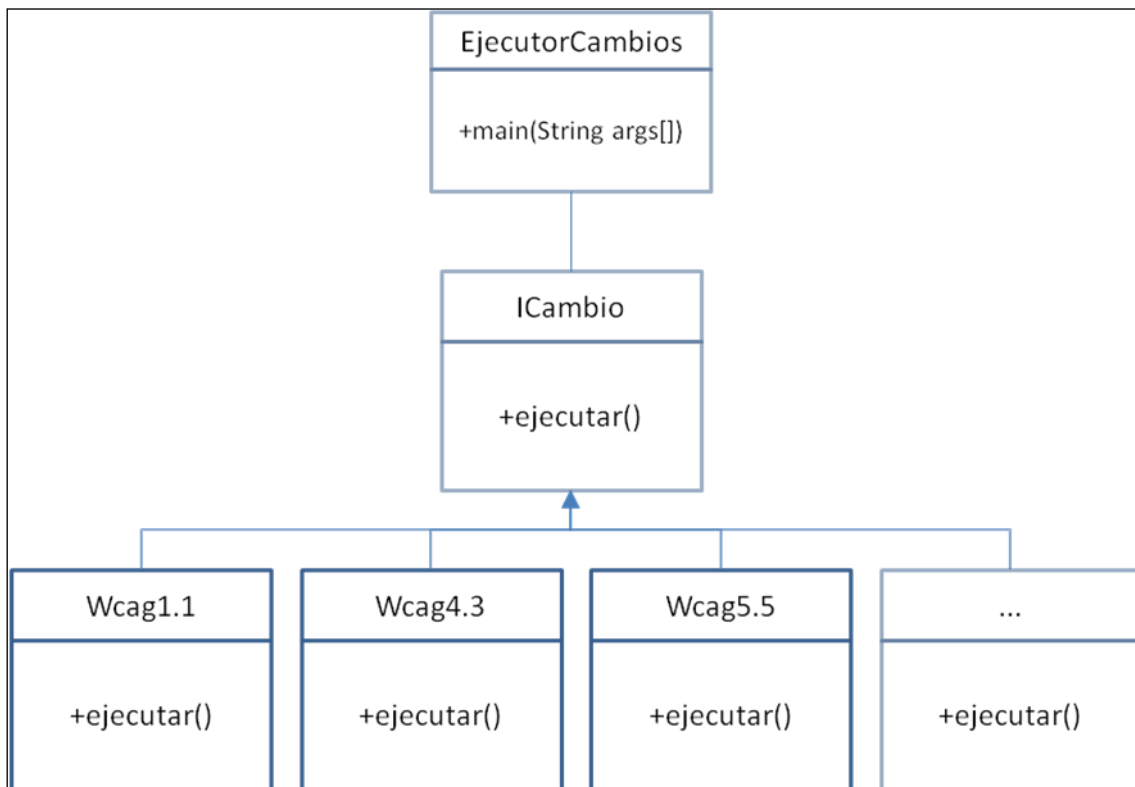


Figura 22 Diagrama de clases de la librería de cambios

De esta forma todo se ejecuta siguiendo un mismo modelo. La función principal del programa, *main*, ejecutará cada una de las clases modificadoras que implementan a la interfaz *ICambio*, llamando al método *ejecutar()*. Esto hace que el sistema se ejecute de forma transparente, ya que todos los cambios se aplican mediante un recorrido del HTML [7] e intercambiando sus propios nodos por los nuevos, que fueron el resultado de la modificación. Además, la ventaja de esta decisión es que incluir un nuevo modificador de cambios es tan simple como añadir una nueva clase que implemente a *ICambio*. Si para el nuevo módulo también se precisa entrada de datos del usuario para el cambio, habrá que añadir una clase al módulo de lógica del sistema, del mismo modo que se añade a esta estructura.

Para este proyecto, se ha implementado una clase por cada tipo de regla que se permite modificar.

5.4.3. Extensión de Mozilla Firefox

La extensión de Mozilla Firefox se decidió implementar como forma de entrada al sistema con el fin de soportar las cookies que se utilicen para validar en un sitio Web. Ésta nos permite acceder a todas las cookies que están almacenadas en el cliente y no sólo a las de nuestro propio dominio. De esta forma se tendrán las cookies que haya almacenadas y que son necesarias para acceder a la página que el usuario pide.

Será por tanto necesario que el usuario se haya validado en el sitio Web antes de intentar lanzar el validador contra una página del sitio, y acceder a WAEX [1] a través de la extensión.

La extensión tiene por nombre WAEX2 y su formato de instalación es el que utiliza Mozilla Firefox en el resto de sus complementos, es decir, **xpi**. Este instalador creará un botón en la barra de herramientas del navegador que al ser pulsado por el usuario abrirá una nueva ventana con el formulario de petición de los datos para la validación de un sitio. La interfaz de usuario que ejecuta el complemento se diseñará como se describió en los prototipos en el punto 5.1 y como se muestra en la Figura 23.

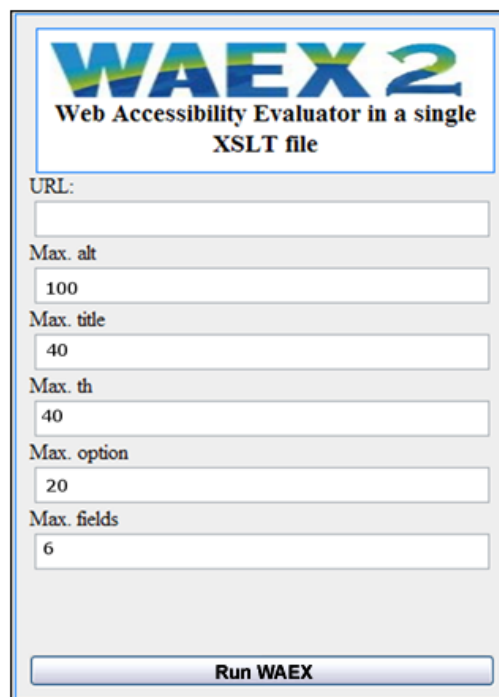


Figura 23 Interfaz de entrada a partir de la extensión de Mozilla Firefox

El envío de los datos se realizará a partir del botón VALIDAR. Es aquí cuando toma gran importancia la funcionalidad de WAEX2 quedando oculta al usuario. Antes de enviar la petición a la página inicial del validador, para su posterior análisis, WAEX [1] extrae las cookies que están almacenadas en el navegador, utilizando para ello una función muy similar a cómo las gestiona éste internamente. Éstas son almacenadas en un fichero en local y posteriormente se envía al servidor, para ser almacenado en el directorio creado únicamente para este usuario. El fichero creado para las cookies tendrá el nombre “cookies.txt” y se escribirá una línea por cada cookie. El formato de cada una de las líneas será el mismo que el utilizado en el navegador Netscape, compatible con el comando WGET [18], y cuya sintaxis es la siguiente:

<Nombre> <Valor> <Expiración> <Ruta> <Dominio> <Secure>

El significado de cada uno de los campos de la definición anterior es el siguiente:

- Nombre: Nombre de la cookie.
- Valor: Valor que se asocia con la cookie.
- Expiración: Fecha en la que la cookie deja de tener validez.
- Ruta: Ruta relativa dentro del host donde tiene validez la cookie.
- Dominio: Host donde tiene validez.
- Secure: Indica si la cookie necesita ser enviada por un medio seguro.

Una vez enviado el fichero de cookies se lanza una petición al servidor donde está instalada la aplicación, con todos los datos que se recogen del usuario y son enviados al servidor mediante cookies.

5.5. Pruebas de funcionamiento

Se ha diseñado un conjunto de pruebas con el fin de revisar y garantizar el buen funcionamiento del software desarrollado. Antes de la presentación del proyecto, el programa implementado ha sido capaz de pasar cada una de las pruebas que se definen a continuación. Entre este conjunto de pruebas se ha dedicado al menos una de ellas por cada una de las funcionalidades que se describían entre los objetivos.

Para la descripción de cada una de las pruebas que han sido validadas, se ha decidido utilizar la siguiente tabla con la descripción de los atributos importantes por cada prueba:

Prueba: P-XX	
Objetivo	
Medio	
Salida esperada	

En la parte superior de la tabla aparece el identificador de la prueba que se ha realizado, con la nomenclatura P-XX donde XX será un número de dos dígitos único para cada prueba.

En la fila de objetivo se describirá qué funcionalidad está comprobando dicha prueba. En la fila con cabecera *Medio* se describirá el procedimiento que se debe seguir para realizar la prueba.

La última fila describirá que debe ocurrir para que se considere que el programa ha pasado la prueba.

5.5.1. Recorrido de páginas

Prueba: P-01	
Objetivo	Dada una página con enlaces, la aplicación será capaz de obtener una lista con todos los enlaces de dicha página.
Medio	Se proporcionará un fichero en formato HTML con mínimo 2 enlaces definidos mediante etiquetas <a>. Estos enlaces creados llevarán a páginas con contenido vacío o sin enlaces.
Salida esperada	La salida esperada será la lista de todos los enlaces de la primera página, sin repetición de ninguno de ellos.

Tabla 36 Prueba P-01

Prueba: P-02	
Objetivo	Dada una página con enlaces, la aplicación será capaz de obtener una lista con todos los enlaces de dicha página y los enlaces a donde llevan estas páginas.
Medio	Se proporcionará como entrada al sistema una página HTML con una serie de enlaces. Cada uno de estos enlaces llevará a otra página con dos enlaces (repetidos o no) cada una.
Salida esperada	La salida del sistema será un listado de todos los enlaces encontrados en estas páginas sin repetición de ninguno de ellos.

Tabla 37 Prueba P-02

Prueba: P-03	
Objetivo	Obtener un listado de páginas donde sus enlaces llevan a ciclos entre ellos.
Medio	La entrada al sistema será un fichero HTML con un enlace que llevará a una nueva página con un enlace a la primera página.
Salida esperada	La salida deberá ser la lista únicamente de los enlaces a las dos páginas.

Tabla 38 Prueba P-03

Prueba: P-04	
Objetivo	Recorrido de todos los elementos que proporcionan una URL.
Medio	<p>La entrada al sistema será una página HTML que lleve a otras (en cualquier orden e incluso con repetición de enlaces) que entre todas ellas contengan al menos uno de cada elemento de la siguiente lista:</p> <ul style="list-style-type: none">• <a>• • <area>• <frame>
Salida esperada	La salida del sistema será el conjunto de todas las URL que componen este conjunto de páginas, y que están incluidos en los elementos anteriores.

Tabla 39 Prueba P-04

5.5.2. Pruebas de validación

Prueba: P-05	
Objetivo	Obtener la validación WAEX [1] de cualquiera de las páginas de una lista.
Medio	Desde la interfaz de validación, seleccionar una página de la lista de enlaces y obtener la validación WAEX [1]. La pestaña de WAEX [1] debe ser la activa.
Salida esperada	Resultado de validación similar al proporcionado por WAEX [1] para la misma página.

Tabla 40 Prueba P-05

Prueba: P-06	
Objetivo	Obtener la validación de color de cualquiera de las páginas de una lista.
Medio	Para cada uno de los enlaces de la lista en la interfaz de validación, seleccionarlos para obtener la validación de color de cada una de ellas. La pestaña Color debe ser la activa.
Salida esperada	El resultado de la validación debe ser un conjunto de todos los elementos cuyos colores no siguen la norma de la W3C [3], o una imagen de la página si es correcta.

Tabla 41 Prueba P-06

5.5.3. Pruebas de soporte a cambios

Prueba: P-07	
Objetivo	El sistema da soporte a cambios sobre WCAG1.1.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG1.1
Salida esperada	El resultado mostrará el error que indica la regla WCAG1.1 y un elemento de entrada de texto para dar opción al usuario a proporcionar texto alternativo.

Tabla 42 Prueba P-07

Prueba: P-08	
Objetivo	El sistema da soporte a cambios sobre WCAG1.2.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG1.2
Salida esperada	El resultado mostrará el error que indica la regla WCAG1.2 y un elemento de entrada de texto para dar opción al usuario a proporcionar texto alternativo.

Tabla 43 Prueba P-08

Prueba: P-09	
Objetivo	El sistema da soporte a cambios sobre WCAG4.3.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG4.3
Salida esperada	El resultado mostrará el error que indica la regla WCAG4.3 y un elemento de selección del idioma para dicha página.

Tabla 44 Prueba P-09

Prueba: P-10	
Objetivo	El sistema da soporte a cambios sobre WCAG5.5.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG5.5
Salida esperada	El resultado mostrará el error que indica la regla WCAG5.5 y un elemento de entrada de texto para dar opción al usuario a proporcionar texto alternativo de la tabla.

Tabla 45 Prueba P-10

Prueba: P-11	
Objetivo	El sistema da soporte a cambios sobre WCAG12.1.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG12.1
Salida esperada	El resultado de la validación mostrará un elemento de texto para dar la opción al usuario a introducir el título del elemento que no se encuentra en el código HTML.

Tabla 46 Prueba P-11

Prueba: P-12	
Objetivo	El sistema da soporte a cambios sobre WCAG13.1.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG13.1
Salida esperada	El resultado de la validación mostrará un elemento de texto para dar la opción al usuario a introducir el texto para el elemento que no se encuentra en el código HTML.

Tabla 47 Prueba P-12

Prueba: P-13	
Objetivo	El sistema da soporte a cambios sobre los elementos <i>font</i> .
Medio	La entrada al sistema será una página HTML que contiene elementos FONT en su código.
Salida esperada	El resultado de la validación mostrará una opción para permitir el cambio de estos elementos por elementos <i>span</i> a través de un <i>checkbox</i> (si está seleccionado se realizará el cambio, sino no)

Tabla 48 Prueba P-13

5.5.4. Pruebas de cambios

Prueba: P-14	
Objetivo	El usuario es capaz de solventar un error producido por la regla WCAG1.1.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG1.1. Se debe introducir un texto para solucionar esta regla. Una vez introducido se deben enviar los cambios haciendo click en el botón GUARDAR CAMBIOS
Salida esperada	El resultado será el mismo que antes salvo que no aparecerá el error para el que se ha proporcionado un texto alternativo.

Tabla 49 Prueba P-14

Prueba: P-15	
Objetivo	El usuario es capaz de solventar un error producido por la regla WCAG1.2.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG1.2. Se debe introducir un texto para solucionar esta regla. Una vez introducido se deben enviar los cambios haciendo click en el botón GUARDAR CAMBIOS
Salida esperada	El resultado será el mismo que antes salvo que no aparecerá el error para el que se ha proporcionado un texto alternativo.

Tabla 50 Prueba P-15

Prueba: P-16	
Objetivo	El usuario es capaz de solventar un error producido por la regla WCAG4.3.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG4.3. Se debe seleccionar un idioma del desplegable que aparece junto al error. Una vez introducido se deben enviar los cambios haciendo click en el botón GUARDAR CAMBIOS
Salida esperada	El resultado será el mismo que antes salvo que no aparecerá el error para el que se ha seleccionado un idioma.

Tabla 51 Prueba P-16

Prueba: P-17	
Objetivo	El usuario es capaz de solventar un error producido por la regla WCAG5.5.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG5.5. Se debe introducir un texto para solucionar esta regla. Una vez introducido se deben enviar los cambios haciendo click en el botón GUARDAR CAMBIOS
Salida esperada	El resultado será el mismo que antes salvo que no aparecerá el error para el que se ha proporcionado un texto alternativo.

Tabla 52 Prueba P-17

Prueba: P-18	
Objetivo	El usuario es capaz de solventar un error producido por la regla WCAG12.1.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG12.1. Se debe introducir un texto para solucionar esta regla. Una vez introducido se deben enviar los cambios haciendo click en el botón GUARDAR CAMBIOS
Salida esperada	El resultado será el mismo que antes salvo que no aparecerá el error para el que se ha proporcionado el texto del elemento que no está presente.

Tabla 53 Prueba P-18

Prueba: P-19	
Objetivo	El usuario es capaz de solventar un error producido por la regla WCAG13.1.
Medio	La entrada al sistema será una página HTML que incumple la regla WCAG13.1. Se debe introducir un texto para solucionar esta regla. Una vez introducido se deben enviar los cambios haciendo click en el botón GUARDAR CAMBIOS
Salida esperada	El resultado será el mismo que antes salvo que no aparecerá el error para el que se ha proporcionado el texto del elemento que no está presente.

Tabla 54 Prueba P-19

5.5.5. Prueba de descarga

Prueba: P-20	
Objetivo	Obtener una copia de las páginas que han pasado por validación siguiendo la misma estructura de directorios en la que se encuentran en su ubicación original.
Medio	Enviar una página para validar que contenga enlaces a otras páginas. Hacer clic en el botón de descargar.
Salida esperada	El resultado será una descarga del resultado comprimido.

Tabla 55 Prueba P-20

6. CAPÍTULO V: CONCLUSIONES Y LÍNEAS FUTURAS

6.1. Conclusiones

Como se decía al comienzo de este documento, la principal meta perseguida en este proyecto era mejorar el validador de accesibilidad Web WAEX [1], consiguiendo resultados que igualen su funcionalidad a la de otros existentes y añadiendo nuevas para facilitar el trabajo al usuario.

Para conseguir estos objetivos se ha tenido que pasar por otros más pequeños, procedentes de una amplia investigación de la situación. Esta investigación ha sido traducida en un análisis y diseño para conseguir el Proyecto de Fin de Carrera que se ha presentado en este documento.

Ya que este proyecto puede ampliarse en nuevas etapas, se ha mostrado con este proyecto una metodología a seguir. También ha sido el primer paso de cara a la mejora de WAEX [1] y sus futuras posibilidades.

Por último, destacar el amplio aprendizaje acerca de la accesibilidad Web y su importancia en el diseño de interfaces de usuario. Además, las normas de accesibilidad, que han sido aprendidas y utilizadas durante el desarrollo del proyecto, no son solo aplicables al Web, sino al desarrollo de las interfaces de cualquier aplicación.

6.2. Líneas futuras

6.2.1. Posibles ampliaciones

La principal motivación de este proyecto era facilitar al usuario el desarrollo de páginas Web accesibles. Dentro de las reglas, existen una gran cantidad de posibilidades para realizar la tarea de modificación de los errores producidos en la validación. Debido a que el proyecto se alargaría demasiado y porque se quería prestar atención a otros puntos que también se podían mejorar en el validador, no se ha decidido dar soporte al cambio de más errores que los estrictamente descritos en este documento. Estos errores que forman parte del proyecto han sido el resultado de seleccionar las de las WCAG 1.0 [4] (la que se consideró desde el principio) aquellas reglas automáticas y sencillas de proporcionar un medio para su solución. El resto, por lo general, eran reglas referentes al contexto, aplicación de estilos o condicionales para las que no había una solución inmediata.

La ampliación de este proyecto, por tanto, podría dirigirse por esta línea. Sería necesario encontrar patrones a través de los cuales se pueda dar soporte a los cambios de los errores que provocan el resto de reglas, tal y como se ha desarrollado en este proyecto con las seleccionadas. La dificultad de encontrar estos patrones se centrará principalmente en las reglas semiautomáticas, que no han sido incluidas en el proyecto. Para estos patrones sería necesario diseñar sistemas especiales que fuesen capaces de identificar errores de un documento HTML [7] que con el actual sistema no es posible. Respecto a las reglas de la WCAG 1.0 [4] se podría estudiar todas aquellas reglas que sean condicionales o dependientes del contexto, como son para reglas como las siguientes:

- Regla 1.3 [11]. Como depende de las posibilidades de los agentes del cliente, sería necesario un sistema que capturase la información auditiva de una presentación multimedia, o proporcionar el mecanismo para ser grabada en el momento.

- Para todas aquellas reglas, y en especial la 3.3, en las que se propone separar el contenido de los documentos HTML [7] de su estilo, sería necesario crear un sistema, un procesador del lenguaje, que detectase donde no se cumplen estas reglas y realizar los cambios oportunos para solventar estos errores. No se propone utilizar un procesador de lenguaje natural, sino más bien uno cuya función sea parecida a la de un compilador, que según reconozca las estructuras HTML sea capaz de separar el contenido del estilo, y este último reescribirlo en una hoja de estilos externa. Se encuentra más información acerca de la creación de este tipo de procesadores de lenguaje en la referencia bibliográfica [30].
- Para la regla 4.1 sería necesario identificar todos los idiomas de los que se compone una página. En este caso, su solución también consistiría en crear un procesador de lenguaje que detectase el idioma primario de cada párrafo del documento y lo asociase en los cambios.
- Otro punto que debe ser mejorado es el tema de los scripts en el lado del cliente. Las reglas WCAG 2.0 [5] es menos estricta referente a este punto pero hay que asegurarse que toda la información a la cuál se accede mediante JAVASCRIPT [20] está también disponible en páginas estáticas. Por ello, sería preferible obligar a que todas las páginas que contengan JAVASCRIPT [20] en su código, tengan también en ellas un enlace a una página con el contenido importante de ésta.

Otra mejora relacionada con estos puntos, sería incorporar un sistema para el reconocimiento automático del lenguaje que predomina en una página. Podría utilizarse para esto un procesador de lenguaje natural, o incluso se podría hacer un clasificador de palabras según el idioma o idiomas a los que podría pertenecer y el lenguaje con más palabras sería el predominante de esa página. A pesar de que estas posibilidades darían solución a la falta de idioma, sería necesario que el usuario confirmase este lenguaje debido a que puede que una página tenga un gran texto en un idioma que no es predominante o natural de la página. Google tiene publicada una herramienta para traducir textos. Además, es capaz de detectar el idioma original del texto. Se puede ver

la información de esta herramienta en la referencia [24]. El problema que presenta esta herramienta, es que aunque permite ser incorporada como traductor en cualquier página web, no permite obtener el reconocimiento de idioma como programa embebido en la web ya que es necesario proporcionar un idioma en el que se da el texto a traducir.

Por otro lado existen programas para el reconocimiento del idioma, que están basados en que cada idioma tiene una frecuencia de ocurrencias de todas las letras, parejas de letras o incluso tríos. En base a esto, se podría determinar el idioma de una página Web. En la referencia [25] se puede ver información de cómo funcionan estos sistemas y el ejemplo de uno en java. Para este proyecto podría implementarse el mismo utilizando PHP o C++.

Evidentemente, el desarrollo de procesadores de lenguaje o sistemas de grabación de voz llevaría un coste de tiempo añadido muy alto y no sería viable para formar parte de un único proyecto de fin de carrera.

Otro punto importante a mejorar, y muy relacionado con el anterior, es el uso de hojas de estilo, aunque este tema es complicado y requiere un amplio estudio de todas las posibilidades que pueden darse en el formato de una página. Se trataría de separar en todo lo posible la estructura del contenido, aunque es complicado conocer exactamente el aspecto de la página de forma automática. Este tipo de correcciones no son triviales, por lo que en el proyecto solo se han tratado los puntos más sencillos, como son los que dan soporte a los errores de color y el cambio de los elementos FONT por SPAN.

6.2.2. Alternativas de implementación

En este punto se intentará describir con el mayor detalle posible cuánto costaría modificar la aplicación para conseguir el mismo funcionamiento sin necesidad de ejecución en un cliente y un servidor como ocurre con la actual. La importancia de este punto es conocer las alternativas para un posible cambio en el futuro, y obtener una aplicación instalable y autónoma, es decir, no se necesitaría ningún otro servidor.

Para comenzar con este análisis, se deberá conocer qué se ejecuta en el servidor y lo qué se ejecuta en el cliente, así como sus lenguajes utilizados.

Actualmente, todo lo que ocurre en el cliente es básicamente la interfaz que comunica el sistema con el usuario. El usuario accede a la aplicación mediante esta interfaz gráfica, desarrollada en HTML [7] y CSS [9]. Para la interacción con los distintos elementos se ha utilizado además JAVASCRIPT [20].

Por otro lado, en el servidor, se ejecuta toda la lógica de la aplicación. Todas las operaciones para obtener la validación tanto de WAEX [1] como la de color se han desarrollado en PHP. Cada vez que el cliente manda una petición de validación llega al servidor los datos mediante POST y son analizados por este código. Además, desde el cliente, también se puede realizar una petición para ejecutar los cambios de mejora de la validación, que es enviada nuevamente al servidor, éste transforma los datos enviados por POST en un fichero XML y es enviado al módulo de cambios que se ejecuta también en el lado del servidor. Este módulo está desarrollado por completo en c++ [23].

Como resumen a este análisis obtenemos la Tabla 56, dividiendo claramente qué se ejecuta en el cliente y qué en el servidor.

	CLIENTE	SERVIDOR
Funcionalidades principales	Extensión de Mozilla Firefox. Interacción CLIENTE-SERVIDOR	Validación de las páginas. Petición de validación a WAEX y al validador de color. Preparación de la salida para realizar los cambios. Ejecución de los cambios.
Herramientas	HTML [7] CSS [9] JAVASCRIPT [20] XUL [21]	PHP C++

Tabla 56 Análisis CLIENTE-SERVIDOR

Con esta información, una nueva aplicación que se ejecutase únicamente en el cliente y con la misma funcionalidad debería tener, por un lado, una interfaz gráfica (que a partir de este momento denominaremos GUI), que constituirá la extensión firefox y será desarrollada en el lenguaje XUL y realizará las acciones necesarias a través de JAVASCRIPT. Por otro lado estará toda la lógica del validador implementada en C++ y que una vez compilada, se ejecutará mediante la llamada a su propio ejecutable. La estructura de la nueva aplicación por tanto quedaría dividida en dos partes fundamentales, tal y como se muestra en la Figura 24:

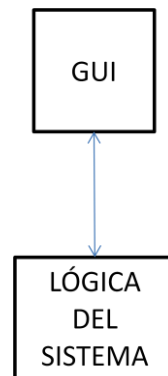


Figura 24 Estructura de mejora de la aplicación

En el GUI únicamente sería necesario implementar las interfaces parecidas a las que se muestran en la aplicación Web, esta vez todo incluido en la misma extensión Mozilla Firefox. Desde la extensión de Mozilla Firefox será posible llamar al ejecutable con la lógica del programa, que devolverá los resultados de validación y cambios utilizando objetos ActiveX. Es necesario utilizar ActiveX ya que JAVASCRIPT [20] no permite lanzar ejecutables por motivos de seguridad. El código para cada llamada será como en el código de la página siguiente:

```
try{  
    var obj=new ActiveXObject("Shell.Application");  
    obj.ShellExecute("C:\\windows\\system32\\calc.exe");  
    return true;  
}catch(e){  
    return false;  
}
```

La lógica del sistema, para atender las peticiones debería ser muy parecida técnicamente. El GUI llamará a la lógica del sistema directamente a través de un ejecutable. Éste se encargará de emular las mismas funciones que se realizaban anteriormente a través del comando GET [17] o WGET [18]. La mayor diferencia en este caso es que debe ser implementado la evaluación de las páginas en el mismo programa C++, sin necesidad que el usuario disponga de conexión a internet. A modo de resumen hasta este punto, el GUI lanzará la petición de validación de una página o listado de páginas proporcionadas por el usuario, y el programa en C++ será el encargado de recoger la petición y analizar a través de un validador interno.

Para la lógica del sistema, como ya se ha dicho anteriormente, el mayor peso de cambio recaería sobre la validación, pues ya no se pueden realizar las peticiones WEB a los validadores existentes. La lógica tendrá como funciones principales, validación WAEX, validación de color, análisis del resultado de la validación y ejecutor de cambios (que ya está implementado en C++). La validación WAEX simplemente consistirá en pasar el fichero wcag.xml [2] y obtendríamos el mismo resultado que en el caso de la petición WEB. Para la validación de la regla 2.2 [12] referente a los colores utilizados, es necesario implementar esta validación para poder obtener una aplicación totalmente independiente. Esta validación consistirá en analizar todas las combinaciones de colores de fuente y fondo utilizadas en la página solicitada. Este proceso será el más costoso en tiempo de esta nueva aplicación.

El coste de implementar esta aplicación sería bastante inferior a la del desarrollo de este proyecto, ya que partiendo de este punto, pasar el código de php a c++ es bastante simple. El problema que se plantea es la validación de color que habrá que tomar decisiones en cuanto a qué colores serán sometidos a validación e incluso analizar

imágenes que han sido fijadas como fondo. Para este segundo caso, podría investigarse la solución de la extensión para Mozilla Firefox para realizar capturas de pantallas, Pearl Crescent Page Saver [26]. Ya que se partirá de una extensión de Mozilla Firefox, se podría obtener esta captura y ser analizada por grupos de pixels para obtener los pares de colores que no son accesibles. En lo que más se estima tardar sería en la realización de las interfaces gráficas del GUI.

En este ejemplo hay que tener en cuenta toda la lógica del programa, de la que se debe separar la librería de cambios que ya está realizada en c++ [23]. Para el resto, habría que dividirlo en las partes fundamentales: implementación de la validación de color, el reconocimiento de texto para las reglas implementadas, interfaces y el resto.

Se comenzará con el análisis de tiempo de migrar las reglas implementadas. Para cada una de las reglas habría que tratar el código php que sustituye un error reconocido por un texto, una entrada de datos y un identificador del error. Lo más lógico sería copiar todos los errores en un array en vez de manejarlos por identificadores como ocurría en php. En el ANEXO I del documento puede verse el código de una regla, en concreto la wcag1.1 [11], desarrollado en PHP. Para esta regla, se ha escrito también el código en c++ [23] para conocer cual sería el coste del paso de un lenguaje a otro. El código de la regla en c++ [23] puede ser consultado en el ANEXO II. El tiempo que se ha utilizado para modificar la regla de ejemplo es de 4.5 horas. Ya que este proyecto está constituido por 6 reglas, el tiempo estimado para esta parte sería de 27 horas.

Para la implementación de las reglas de validación de color sería el proceso con mayor coste en tiempo. Sería necesaria en primer lugar una previa investigación de las diferentes posibilidades de reconocimiento de color en las páginas, ya explicadas anteriormente. Esta investigación se estima que debería durar 32 horas por cada una de las posibilidades descritas como:

- A través de la extensión que recoge la imagen de la página [26]. Consistiría en el análisis de los colores de la imagen que guarda la extensión y someter éstos a las reglas de validación de color.

- A partir de la recogida de todos los colores que aparecen en el código y ficheros CSS asociados a un documento, y clasificados éstos, según su uso, en color de fuente y color de fondo. Aplicación de las reglas de validación de color a cada par de colores entre estos dos grupos.

Estas investigaciones, se da tiempo suficiente a la obtención de todo el material necesario para la posterior implementación. Tras la investigación de las dos posibilidades, se debería realizar un análisis de cuál sería la mejor solución. Este análisis se estima que podría estar completo, con documentación incluida, en 16 horas. Una vez concluido el análisis se procedería a la implementación, que dada la dificultad de selección de los colores de la página, se estima en 30 horas.

En total, la implementación de la nueva validación de la regla de color, estaría estimada en 110 horas. Son muchas horas añadidas pero nos aseguramos de la total independencia del programa respecto a otros.

Las interfaces principales del programa junto con la estimación de las horas que llevaría su implementación se muestra en la Tabla 57:

INTERFAZ	TIEMPO ESTIMADO
Entrada	3 hora
Validación de WAEX	5 horas
Validación de Color	5 horas

Tabla 57 Estimación de horas para interfaces

Por tanto el tiempo total estimado para la creación de las interfaces es de 13 horas.

En cuanto al resto de lógica, y debido a que está implementado en PHP, es decir, será muy similar a c++ [23], la estimación del tiempo se hará en base a las líneas de código que comprende. El código en PHP que se está analizando tiene 2555. El número de líneas de código de la regla que ha sido implementada en c++ [23] como ejemplo tiene

250 líneas. Puesto que anteriormente se ha estimado 4.5 horas por cada 250 línea de código, para esta parte se creen necesarias aproximadamente 46 horas.

Se debe pensar también, en que será necesaria contar con un pequeño margen temporal para imprevistos a la hora de implementar en c++ [23]. Se considerará por tanto un tiempo de error de 8 horas.

Puesto que el cambio de un lenguaje a otro implicará que la aplicación pase de ser ejecutada en el servidor a ser ejecutada en local, y su funcionamiento interno será por tanto diferente, se considera necesario un amplio periodo dedicado a pruebas y fallos del sistema. Este periodo debería ser la mitad del tiempo estimado para su desarrollo, es decir, lo considero suficiente con 23 horas.

La implementación de este nuevo sistema necesitaría documentación acerca del nuevo manual de usuario y un nuevo manual de instalación. Este último solo consistiría en la explicación de cómo instalar la extensión Mozilla Firefox, ya que ésta misma incluiría el ejecutable del programa C++ que se encarga de todo el trabajo de validación. El tiempo para estos dos manuales se estima en 8 horas.

En resumen, en la siguiente tabla (Tabla 58) se muestra todos los tiempos que se han detallado anteriormente y el tiempo final estimado para el cambio de una implementación en PHP a c++ [23].

CONCEPTO	TIEMPO (en Horas)
Implementación de las reglas	27
Implementación de las reglas de color	110
Interfaces	13
Resto de código	46
Imprevistos	8
Documentación (manuales)	8
Prueba y error	23
Total	235

Tabla 58 Estimación de horas para C++

Por tanto, se estima un total de 235 horas para implementar el proyecto en C++ y la extensión Mozilla Firefox. Estas horas podrían traducirse en días laborables (considerando 8 horas por día), de aproximadamente 29 días. Considerando este tiempo en un proyecto real, una empresa necesitaría de 29 días laborables, es decir, aproximadamente 6 semanas de trabajo.

Esta hubiera sido una posible solución al problema inicial, o incluso una ampliación futura a este proyecto. Si el análisis de ambas soluciones se centrara únicamente en cuál de ellas es mejor en relación a la meta que persiguen, es decir, cuál de ellas es más accesible, se puede destacar que la implementada cumple:

- Al ser una aplicación Web hace que sea multiplataforma, ya que no depende del sistema operativo.
- No es necesario una instalación previa a su uso por parte del usuario, por lo que no necesita de conocimientos más avanzados que los de un usuario habitual del ordenador.

Ya que la meta que se quiere perseguir con el software que se ha desarrollado es promover la creación de páginas Web accesibles, creo que son suficientes motivos para afirmar que ésta solución es mejor.

Sin embargo, la aplicación que se propone como alternativa presentaría mejoras en cuanto a que no necesitaría un servidor externo al que conectarse para ejecutar la aplicación, y las respuestas proporcionadas por éste serían más rápidas, dependiendo esta diferencia de tiempo del tipo de conexión y ancho de banda del usuario en el caso de la aplicación Web.

6.2.3. WCAG 2.0

Como se ha mencionado en puntos anteriores, en la actualidad, la accesibilidad web debería comprobarse mediante las reglas que se publican por la W3C [3] en su segunda versión, es decir, las WCAG 2.0 [12]. La razón por la que este proyecto está basado en la primera versión es porque estas reglas han sido incorporadas al estándar

posteriormente al inicio de este proyecto. A continuación, se van presentar las diferencias entre las dos versiones de un modo adaptado a la filosofía de este proyecto, es decir, todas aquellas reglas que sean nuevas y que se puedan añadir en la mejora del proyecto.

Estas diferencias relevantes al proyecto son:

- Para la regla 1.1 de la primera versión, se identifican los elementos que no son de tipo texto de los cuales se podría añadir a la regla de modificación en la post-validación los inputs, controls y elementos multimedia.
- En cuanto al control de uso de color, la segunda versión permite mayor libertad respetando un contraste mínimo 4.5:1 o si el texto tiene una fuente grande, es información sin importancia en la página o se utiliza el texto sobre logos. Debido a que estas últimas características no son de comprobación automática, la regla que debe seguir en el validador es conseguir un contraste mínimo de 4.5:1.
- Se debería intentar conseguir el uso máximo de tamaños absolutos respecto a relativos. Con esta regla, se quiere decir, que todos los tamaños utilizados, por ejemplo en las fuentes, sean relativos a la letra seleccionada en el navegador y no con valores absolutos. Para dar soporte a este error sería necesario diseñar un mecanismo para que el usuario pudiese cambiar, tras la validación, todos los tamaños absolutos por otros relativos y ver los cambios con una previa visualización.
- En la primera versión, se pretende evitar todos los mecanismos de refresco y uso de javascript [20], a menos que existan mecanismos para acceder a la información evitando estos elementos. La herramienta que se ha implementado en este proyecto no da soporte a javascript [20], por lo que estas reglas podrían ser demasiado tedioso incorporarlas al validador. Además, por lo general no son reglas comprobables de forma automática.

ANEXO I Código de una regla en PHP

Código en PHP de la regla 1.1

```
<?php
```

```
class WCAG1_1  
{
```

```
    var $ids;  
    var $pagina_errores;  
    var $pagina_real;  
    var $css;  
    var $numErrores;
```

```
// method declaration
```

```
function WCAG1_1($pagErrores, $pagReal, $cssReal){
```

```
    if( $pagErrores!=""){  
        $this->ids = array();  
        $this->setPaginaErrores($pagErrores);  
        $this->setPaginaReal($pagReal);  
        $this->setCssReal($cssReal);  
        $this->numErrores = 0;  
    }  
    else{  
        $pagina_real = $pagReal;  
    }  
}
```

```
}
```

```
function solucionarProblemas($post, $pagina){  
    $numErrores = $post["numImg"];
```

```
    $paginaRelativa = str_replace("http://www.gast.it.uc3m.es/~salmodovar/", "/",$pagina);
```

```
    $fd = fopen($paginaRelativa, "r");  
    $paginaEntera = "";  
    while(!feof($fd)){  
        $linea = fgets($fd);  
        echo $linea;  
        $paginaEntera = $paginaEntera . $linea;  
    }  
    fclose($fd);
```

```
    for( $i=0; $i < $numErrores; $i++){  
        $indice = "imgAlt$i";  
        $imgAlt = $post[$indice];  
  
        $indice = "textAlt$i";  
        $textAlt = $post[$indice];  
        $textAlt = str_replace("\\\\", "\\", $textAlt);  
        $sustituta = $textAlt;
```

```
        $sustituta = str_replace("</>", " alt=\"\".\"$imgAlt.\"\" />", $sustituta);
```

```
        $paginaEntera = str_replace($textAlt, $sustituta,$paginaEntera);
```

```
    }

    $fd = fopen($paginaRelativa, "w");
    fwrite($fd,$paginaEntera);
    fclose($fd);

    echo $paginaEntera;
}

//function set
function setPaginaErrores($pag){
    $this->pagina_errores = $pag;
}
function setPaginaReal($pag){
    $this->pagina_real = $pag;
}

function setCssReal($pag){
    $this->css = $pag;
}

function getPaginaErrores(){
    return $this->pagina_errores;
}
function getPaginaReal(){
    return $this->pagina_real;
}

function getCssReal(){
    return $this->css;
}

function parser() {

    $this->parserImg();
    $this->parserArea();
    $this->parserInputs();

    $this->pagina_errores = $this->pagina_errores . "<input type='hidden' name='numImg' value='$this-
>numErrores'>";
}

function parserImg(){
    $titulo = "Images with no alt attribute";
    $pos = strpos($this->pagina_errores,$titulo);

    $max = strpos($this->pagina_errores,"</ul>",$pos);

    if ( !( strpos($this->pagina_errores,$titulo) === FALSE ) ){
        //hay al menos un error de este tipo
        $ini_p = strpos($this->pagina_errores, "<ul>", $pos);
        $fin_p = strpos($this->pagina_errores, "</ul>", $pos) - $ini_p;

        $lista = substr($this->pagina_errores, $ini_p, $fin_p);
        $ini = 0;
        while ($ini < $max && !( strpos($lista, "<li>", $ini) === FALSE) ){
            $ini = strpos($lista, "<li>", $ini);
            $fin = strpos($lista, "</li>", $ini)-$ini;
            $anterior = substr($lista, $ini, $fin);
```

```
$nueva = str_replace("<samp", "<samp id='imgTexto$this->numErrores' ", $anterior);

$ini_samp = strpos( $lista, ">", $ini+4)+1;
$fin_samp = strpos( $lista, "</samp>", $ini_samp)-$ini_samp;
$texto = substr( $lista, $ini_samp, $fin_samp);

$ini_imagen = strpos($lista, "<samp>") + 6;
$fin_imagen = strpos($lista, "</samp>")-$ini_imagen;
$imagen = substr($lista, $ini_imagen, $fin_imagen);
echo "la imagen recogida es: $imagen<br>";

$nueva = str_replace("</samp>", " <br>Description for this image: <input type='text'
name='imgAlt$this->numErrores'></samp>", $nueva);
$nueva = str_replace("</samp>", "<input type='hidden' name='textAlt$this-
>numErrores' value='$texto'></samp>", $nueva);

$this->pagina_errores = substr($this->pagina_errores,0,$ini_p+$ini) . $nueva .
substr($this->pagina_errores,$ini_p+$ini+$fin);

$ini = $ini + $fin+1;
$ini_p = strpos($this->pagina_errores, "<ul>", $pos);
$fin_p = strpos($this->pagina_errores, "</ul>", $pos) - $ini_p;
$lista = substr($this->pagina_errores, $ini_p, $fin_p);
$this->numErrores++;
    }
}

function parserArea(){
    $titulo = "Areas with no alt attribute";
    $pos = strpos($this->pagina_errores,$titulo);
    $max = strpos($this->pagina_errores,"</ul>",$pos);

    if ( !( strpos($this->pagina_errores,$titulo) === FALSE ) ){
        //hay al menos un error de este tipo
        $ini_p = strpos($this->pagina_errores, "<ul>", $pos);
        $fin_p = strpos($this->pagina_errores, "</ul>", $pos) - $ini_p;

        $lista = substr($this->pagina_errores, $ini_p, $fin_p);
        $ini = 0;
        while ( $ini < $max && !( strpos($lista, "<li>", $ini) === FALSE) ){

            $ini = strpos($lista, "<li>", $ini);
            $fin = strpos($lista, "</li>", $ini)-$ini;
            $anterior = substr($lista, $ini, $fin);
            $nueva = str_replace("<samp", "<samp id='imgTexto$this->numErrores'", $anterior);
            $nueva = str_replace("</samp>", " <br>Description for this area: <input type='text'
name='imgAlt$this->numErrores'></samp>", $nueva);

            $ini_samp = strpos( $lista, ">", $ini+4)+1;
            $fin_samp = strpos( $lista, "</samp>", $ini_samp)-$ini_samp;
            $texto = substr( $lista, $ini_samp, $fin_samp);

            $nueva = str_replace("</samp>", " <input type='hidden' name='textAlt$this-
>numErrores' value='$texto'></samp>", $nueva);

            $this->pagina_errores = substr($this->pagina_errores,0,$ini_p+$ini) . $nueva .
substr($this->pagina_errores,$ini_p+$ini+$fin);
            //$this->pagina_errores = str_replace($anterior, $nueva, $this->pagina_errores);
```

```

        $ini = $ini + $fin+1;
        $ini_p = strpos($this->pagina_errores, "<ul>", $pos);
        $fin_p = strpos($this->pagina_errores, "</ul>", $pos) - $ini_p;
        $lista = substr($this->pagina_errores, $ini_p, $fin_p);

        $this->numErrores++;
    }
}

function parserInputs(){
    $titulo = "Image inputs with no alt attribute";
    $pos = strpos($this->pagina_errores,$titulo);
    $max = strpos($this->pagina_errores,"</ul>", $pos);

    if (!( strpos($this->pagina_errores,$titulo) === FALSE )){
        //hay al menos un error de este tipo
        $ini_p = strpos($this->pagina_errores, "<ul>", $pos);
        $fin_p = strpos($this->pagina_errores, "</ul>", $pos) - $ini_p;

        $lista = substr($this->pagina_errores, $ini_p, $fin_p);
        $ini = 0;
        while ( $ini < $max && !( strpos($lista, "<li>", $ini) === FALSE) ){
            $ini = strpos($lista, "<li>", $ini);
            $fin = strpos($lista, "</li>", $ini)-$ini;
            $anterior = substr($lista, $ini, $fin);
            $nueva = str_replace("<samp", "<samp id='imgTexto$this->numErrores'", $anterior);
            $nueva = str_replace("</samp>", " <br>Description for this input element: <input
type='text' name='imgAlt$this->numErrores'></samp>", $nueva);

            $ini_samp = strpos( $lista, ">", $ini+4)+1;
            $fin_samp = strpos( $lista, "</samp>", $ini_samp)-$ini_samp;
            $texto = substr( $lista, $ini_samp, $fin_samp);

            $nueva = str_replace("</samp>", "<input type='hidden' name='textAlt$this-
>numErrores' value='$texto'></samp>", $nueva);

            $this->pagina_errores = substr($this->pagina_errores,0,$ini_p+$ini) . $nueva .
substr($this->pagina_errores,$ini_p+$ini+$fin);
            //$this->pagina_errores = str_replace($anterior, $nueva, $this->pagina_errores);

            $ini = $ini + $fin+1;
            $ini_p = strpos($this->pagina_errores, "<ul>", $pos);
            $fin_p = strpos($this->pagina_errores, "</ul>", $pos) - $ini_p;
            $lista = substr($this->pagina_errores, $ini_p, $fin_p);
            $this->numErrores++;
        }
    }
}

}

?>
```

ANEXO II Código de una regla en C++

Código en C++ de la regla 1.1

```
public class WCAG1_1
{
    char * pagina_errores;
    char *pagina_real;
    char *css;
    int numErrores;

    // method declaration

    public WCAG1_1(char *pagErrores, char *pagReal, char *cssReal){

        if( strcmp(pagErrores,"")==0){
            pagina_errores = (char *)malloc(strlen(pagErrores));
            strcpy(pagina_errores,pagErrores);
            pagina_real = (char *)malloc(strlen(pagReal));
            strcpy(pagina_real, pagReal);
            css = (char *)malloc(strlen(cssReal));
            strcpy(css, cssReal);
            numErrores = 0;
        }
        else{
            pagina_real = (char *)malloc(strlen(pagReal));
            strcpy(pagina_real, pagReal);
        }
    }

    void solucionarProblemas(char **post, char *pagina){
        int numErrores = post[0];
        char *paginaRelativa = "http://www.gast.it.uc3m.es/~salmodovar/";

        if ( pagina[0] == '.' && pagina[1] == '/')
            strcpy(paginaRelativa+strlen(paginaRelativa),pagina+2);
        else
            strcpy(paginaRelativa,pagina);

        FILE *fd = fopen($paginaRelativa, "r");
        char *paginaEntera = (char *)malloc(10000);
        while(fgets(paginaEntera+strlen(paginaEntera), 5000, fd) != NULL);
        fclose($fd);
    }
}
```

```
for( int i=0; int i < numErrores; i++){
    int indice = "imgAlt$i";
    char *imgAlt = post[indice];

    $indice = "textAlt" + i;
    $textAlt = post[indice];
    $sustituta = textAlt;

    $sustituta = replace("/>", " alt=\"\".\"$imgAlt.\"\" />", $sustituta);

    $paginaEntera = replace($textAlt, $sustituta,$paginaEntera);
}
fd = fopen(paginaRelativa, "w");
fwrite(fd,strlen(paginaEntera),1,paginaEntera);
fclose(fd);
}
char * getPaginaErrores(){
    return pagina_errores;
}
char * getPaginaReal(){
    return pagina_real;
}
char* getCssReal(){
    return css;
}

void parser() {
    parserImg();
    parserArea();
    parserInputs();
    strcpy(pagina_errores+strlen(pagina_errores),"<input          type='hidden'          name='numImg'
value='numErrores'>");
}

int strpos( char * str, char *needle, int pos=0){
    char *p = strstr(str,needle,pos);
    if (p) return p-str;
    return -1;
}

char * substr(char * stroriginal, int ini, int fin){

    char * strReturn = (char *)calloc(fin-ini+1);
```

```
strcpy( strReturn, stroriginal+ini, fin);
return strReturn;
}

char * str_replace(char * original, char * old, char *new){

    if ( strstr( original, new) == NULL){
        return NULL;
    }
    char *dest = (char *)calloc(strlen(original)+strlen(new));
    int ini = strpos(original, old,0);
    strcpy( dest, original, ini);
    strcpy( dest + strlen(dest), new, strlen(new) );
    strcpy( dest + strlen(dest), original+ini+strlen(old) );
    return dest;
}

void parserImg(){
    char *titulo = "Images with no alt attribute";
    int pos = strstr(pagina_errores,titulo)-pagina_errores;

    int max = strstr(pagina_errores,"</ul>",pos)-pagina_errores;

    if ( strstr(pagina_errores, titulo) != NULL){

        int ini_p = strpos(pagina_errores, "<ul>", pos);
        int fin_p = strpos(pagina_errores, "</ul>", pos) - ini_p;

        char *lista = substr(pagina_errores, ini_p, fin_p);
        int ini = 0;
        int fin = 0;
        char * anterior;
        char * pattern = (char *)malloc(250);
        while (ini < max && ! strpos(lista, "<li>", ini) >= 0) ){
            ini = strpos(lista, "<li>", ini);
            fin = strpos(lista, "</li>", ini)-ini;
            anterior = substr(lista, ini, fin);
            sprintf(pattern"<samp id='imgTexto%d' ",numErrores);

            char *nueva = str_replace("<samp", pattern, anterior);

            int ini_samp = strpos( lista, ">", ini+4)+1;
            int fin_samp = strpos( lista, "</samp>", ini_samp)-ini_samp;
            char * texto = strstr( lista, ini_samp, fin_samp);
```



```
int ini_imagen = strpos($lista, "<samp>") + 6;
int fin_imagen = strpos($lista, "</samp>")-$ini_imagen;
imagen = substr($lista, $ini_imagen, $fin_imagen);
char * input = (char *)calloc(250);
sprintf(input, "    <br>Description    for    this    image:    <input    type='text'
name='imgAlt%d'></samp>",numErrores);
nueva = str_replace("</samp>", input, $nueva);
nueva = str_replace("</samp>", "<input type='hidden' name='textAlt$this->numErrores'
value='$texto'></samp>", $nueva);

strcpy(pagina_errores, substr(pagina_errores,0,ini_p+ini);
strcpy(pagina_errores+strlen(pagina_errores,nueva);
strcpy(pagina_errores+strlen(pagina_errores, substr(pagina_errores,ini_p+ini+fin));

ini = ini + fin+1;
ini_p = strpos(pagina_errores, "<ul>", pos);
fin_p = strpos(pagina_errores, "</ul>", pos) - ini_p;
lista = substr(pagina_errores, ini_p, fin_p);
numErrores++;
    }
}
}

void parserArea(){
    char *titulo = "Areas with no alt attribute";
    int pos = strstr(pagina_errores,titulo)-pagina_errores;

    int max = strstr(pagina_errores,"</ul>",pos)-pagina_errores;

    if ( strstr(pagina_errores, titulo) != NULL){

        int ini_p = strpos(pagina_errores, "<ul>", pos);
        int fin_p = strpos(pagina_errores, "</ul>", pos) - ini_p;

        char *lista = substr(pagina_errores, ini_p, fin_p);
        int ini = 0;
        int fin = 0;
        char * anterior;
        char * pattern = (char *)malloc(250);
        while (ini < max && ! strpos(lista, "<li>", ini) >= 0 ){
            ini = strpos(lista, "<li>", ini);
            fin = strpos(lista, "</li>", ini)-ini;
            anterior = substr(lista, ini, fin);
            sprintf(pattern"<samp id='imgTexto%d' ",numErrores);
```

```
char *nueva = str_replace("<samp", pattern, anterior);

int ini_samp = strpos( lista, ">", ini+4)+1;
int fin_samp = strpos( lista, "</samp>", ini_samp)-ini_samp;
char * texto = strstr( lista, ini_samp, fin_samp);

int ini_imagen = strpos($lista, "<samp>") + 6;
int fin_imagen = strpos($lista, "</samp>")-$ini_imagen;
imagen = substr($lista, $ini_imagen, $fin_imagen);

char * input = (char *)calloc(250);
sprintf(input, "    <br>Description    for    this    area:    <input    type='text'
name='imgAlt%d'></samp>", numErrores);
nueva = str_replace("</samp>", input, $nueva);
nueva = str_replace("</samp>", "<input type='hidden' name='textAlt$this->numErrores'
value='$texto'></samp>", $nueva);

strcpy(pagina_errores, substr(pagina_errores,0,ini_p+ini);
strcpy(pagina_errores+strlen(pagina_errores,nueva);
strcpy(pagina_errores+strlen(pagina_errores, substr(pagina_errores,ini_p+ini+fin));

ini = ini + fin+1;
ini_p = strpos(pagina_errores, "<ul>", pos);
fin_p = strpos(pagina_errores, "</ul>", pos) - ini_p;
lista = substr(pagina_errores, ini_p, fin_p);
numErrores++;
    }
}
}

void parserInputs(){
    char *titulo = "Image inputs with no alt attribute";
    int pos = strstr(pagina_errores,titulo)-pagina_errores;

    int max = strstr(pagina_errores,"</ul>",pos)-pagina_errores;

    if ( strstr(pagina_errores, titulo) != NULL){

        int ini_p = strpos(pagina_errores, "<ul>", pos);
        int fin_p = strpos(pagina_errores, "</ul>", pos) - ini_p;

        char *lista = substr(pagina_errores, ini_p, fin_p);
        int ini = 0;
        int fin = 0;
```

```
char * anterior;
char * pattern = (char *)malloc(250);
while (ini < max && ! strpos(lista, "<li>", ini) >= 0 ){
    ini = strpos(lista, "<li>", ini);
    fin = strpos(lista, "</li>", ini)-ini;
    anterior = substr(lista, ini, fin);
    sprintf(pattern"<samp id='imgTexto%d' ",numErrores);

    char *nueva = str_replace("<samp", pattern, anterior);

    int ini_samp = strpos( lista, ">", ini+4)+1;
    int fin_samp = strpos( lista, "</samp>", ini_samp)-ini_samp;
    char * texto = strstr( lista, ini_samp, fin_samp);

    int ini_imagen = strpos($lista, "<samp>") + 6;
    int fin_imagen = strpos($lista, "</samp>")-$ini_imagen;
    imagen = substr($lista, $ini_imagen, $fin_imagen);

    char * input = (char *)calloc(250);
    sprintf(input," <br>Description for this input element: <input type='text'
name='imgAlt%d'></samp>",numErrores);
    nueva = str_replace("</samp>", input, $nueva);
    nueva = str_replace("</samp>", "<input type='hidden' name='textAlt'
value='$texto'></samp>", $nueva);

    strcpy(pagina_errores, substr(pagina_errores,0,ini_p+ini);
    strcpy(pagina_errores+strlen(pagina_errores,nueva);
    strcpy(pagina_errores+strlen(pagina_errores, substr(pagina_errores,ini_p+ini+fin));

    ini = ini + fin+1;
    ini_p = strpos(pagina_errores, "<ul>", pos);
    fin_p = strpos(pagina_errores, "</ul>", pos) - ini_p;
    lista = substr(pagina_errores, ini_p, fin_p);
    numErrores++;
}
}
}
}
```

ANEXO III INSTALACIÓN Y CONFIGURACIÓN

1. Instalación y configuración del sistema

1.1.1. Prerrequisitos de instalación

El sistema se compone de una carpeta que será copiada en el servidor de destino. Para poder instalar la aplicación se precisa de un servidor Web que funcione bajo el sistema operativo Linux.

Además es imprescindible que en el servidor esté instalado PHP [22] con una versión 4.4.4 o superior.

1.1.2. Instalación y configuración

Para realizar la instalación del validador únicamente es necesario copiar el contenido de la carpeta con el código del proyecto en la ubicación del servidor, de tal forma que ésta misma carpeta será la ruta de entrada en la aplicación. Es decir, si copiamos el contenido del proyecto en la carpeta raíz de la carpeta *www* donde está funcionando el servidor, cuya URL es *www.uc3m.es*, se podría entrar a la aplicación utilizando la siguiente ruta: *http://www.uc3m.es/index.php4*.

Para que el sistema funcione correctamente, es necesario editar una línea en el fichero *waex.conf* que se acompaña al código del proyecto, y el cual también es necesario copiarlo en el servidor. La línea que hay que modificar es la que contiene la variable *host*, en la que se deberá poner la ruta de acceso que se ha explicado anteriormente. La línea está inicialmente comentada ya que comienza con el carácter *#* por lo que habrá que modificarlo. Resumiendo, hay que cambiar una página que tiene un aspecto como el siguiente por la misma sin el carácter *#* y con tus propios valores de instalación:

```
#install_host = http://my_host/install_folder
```

Con esto, la aplicación queda lista para ser usada vía únicamente Web.

1.1.3. Instalación de la aplicación para cada uno de los usuarios

Para la instalación de la extensión, se deberán seguir los siguientes pasos. Arrastrar el fichero waex.xpi incluido en el paquete de instalación (o descargarlo desde la URL donde esté disponible) hasta una ventana del navegador Mozilla Firefox. Proceder a la instalación del nuevo complemento (Figura 25):

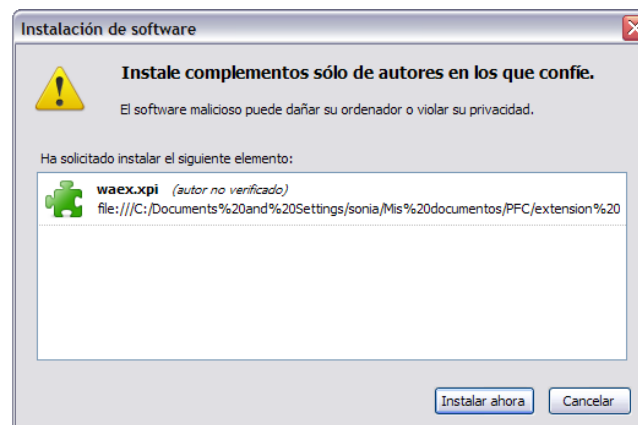


Figura 25 Instalación de la extensión WAEX2 para Mozilla Firefox

Una vez instalado, el mismo navegador pedirá su reinicio. Se deberán cerrar todas las ventanas para que se hagan efectivos los cambios.

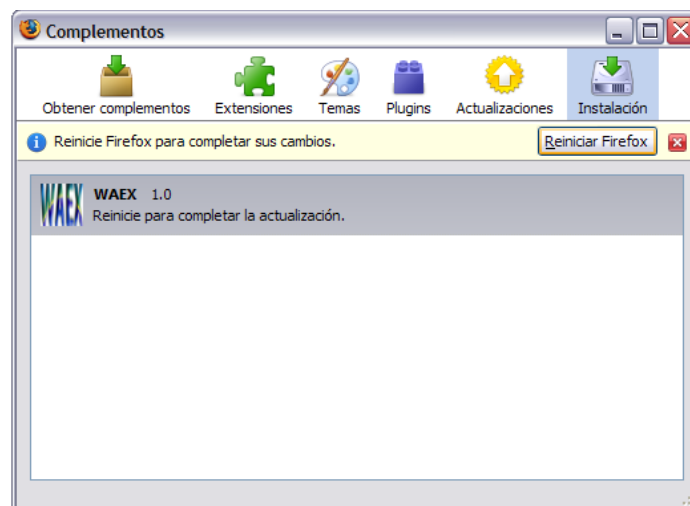


Figura 26 Instalación de la extensión WAEX2 para Mozilla Firefox (completada)

Al reiniciar el navegador, para poder tener acceso en todo momento y de forma rápida a la aplicación, es necesario incorporar el botón de la extensión en la barra de herramientas del navegador. Para ello, se deberá hacer clic con el botón derecho del ratón sobre la parte superior de la barra de herramientas, y presionar sobre **Personalizar**.

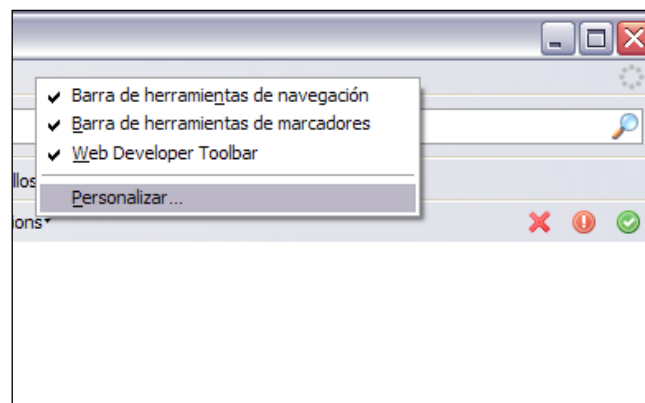


Figura 27 Instalación de la extensión WAEX2 para Mozilla Firefox (1)

Buscar el icono de WAEX [1] y arrastrarlo hasta la posición de la barra de herramientas donde se desea colocar. El icono que debe arrastrarse se muestra en la siguiente imagen (Figura 28):

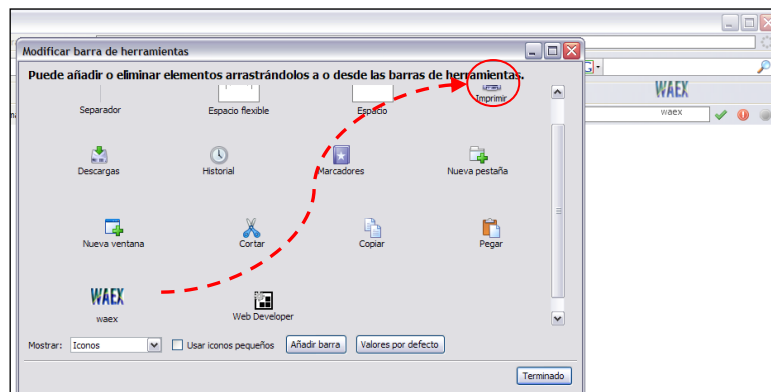


Figura 28 Instalación de la extensión WAEX2 para Mozilla Firefox (2)

La aplicación está lista para ser utilizada.

ANEXO IV DOCUMENTACIÓN

1. Manual de usuario

1.1.1. Introducción a WAEX

WAEX 2 es una aplicación Web cuya utilidad es la validación de un sitio Web. A continuación se explica cómo debe utilizarse en su forma más completa, es decir, a partir de la extensión WAEX 2 que se considera previamente instalada en el navegador Mozilla Firefox.

La entrada al sistema será a través del botón que se ha añadido en la instalación de la extensión, y que estará situado en la barra de herramientas del navegador Mozilla Firefox como se muestra en la Figura 29.

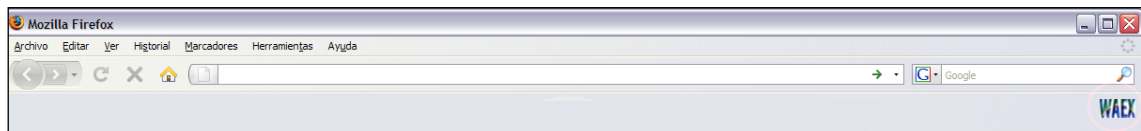


Figura 29 WAEX2 en la barra de herramientas de Mozilla Firefox

Pulsando sobre este botón se abrirá una nueva ventana donde se piden los datos para la validación, tal y como se muestra en la siguiente imagen (Figura 30):

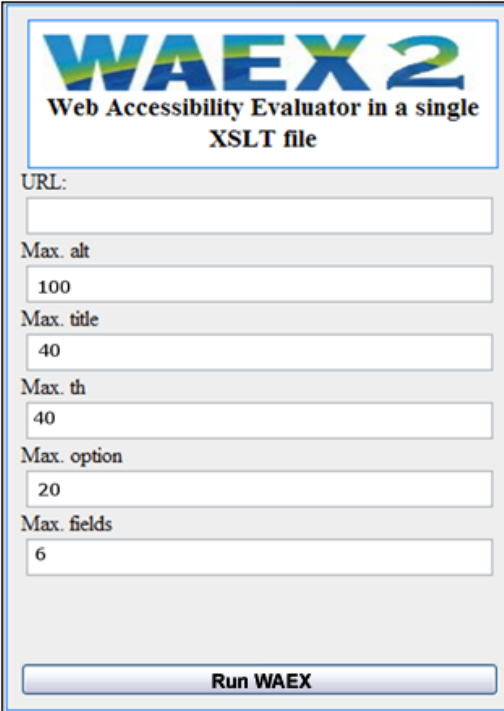


Figura 30 Interfaz de entrada a WAEX2 a través de la extensión

A continuación se describen cada uno de los campos:

URL: Es la dirección donde se encuentra la página a partir de la que se comenzará la validación. Esto significa que aunque sólo se envíe una URL, se podrán validar todas las páginas a las cuales se pueda acceder por navegación de los hipervínculos desde la inicial.

Max. Alt. Es la longitud máxima que puede tener cualquier atributo *alt* que aparezca en los documentos que serán validados con WAEX [1] para que éste no lo considere como un error.

Max. Title: Es la longitud máxima que puede tener cualquier elemento *title* que aparezca en los documentos que serán validados con WAEX [1] para que éste no lo considere como un error.

Max. Th: Longitud máxima para las cabeceras de las tablas que aparecen en cualquier de los documentos que serán validados con WAEX [1] para que éste no lo considere como un error.

Max. Option: Es el máximo número de opciones que pueden aparecer en un campo de un formulario, de cualquiera de los documentos que serán validados con WAEX [1] para que éste no lo considere como un error.

Max. Fields: Es el máximo número de campos que pueden existir agrupados en un formulario que aparezca en los documentos que serán validados con WAEX [1] para que éste no lo considere como un error.

No es necesario rellenar todos los campos ya que WAEX [1] incorpora valores por defecto a aquellos que no se les indique ningún valor. Aunque es recomendable rellenarlos.

Una vez rellenados todos los campos que se desea, se comienza la validación pulsando sobre el botón que aparece en la parte inferior de la ventana. A continuación nos situaremos en una nueva ventana con dos modos de validación distintas, entre las que se podrá cambiar entre una y otra (para validar siempre la página activa en ese momento) a través de los enlaces indicados en la siguiente imagen (Figura 31):

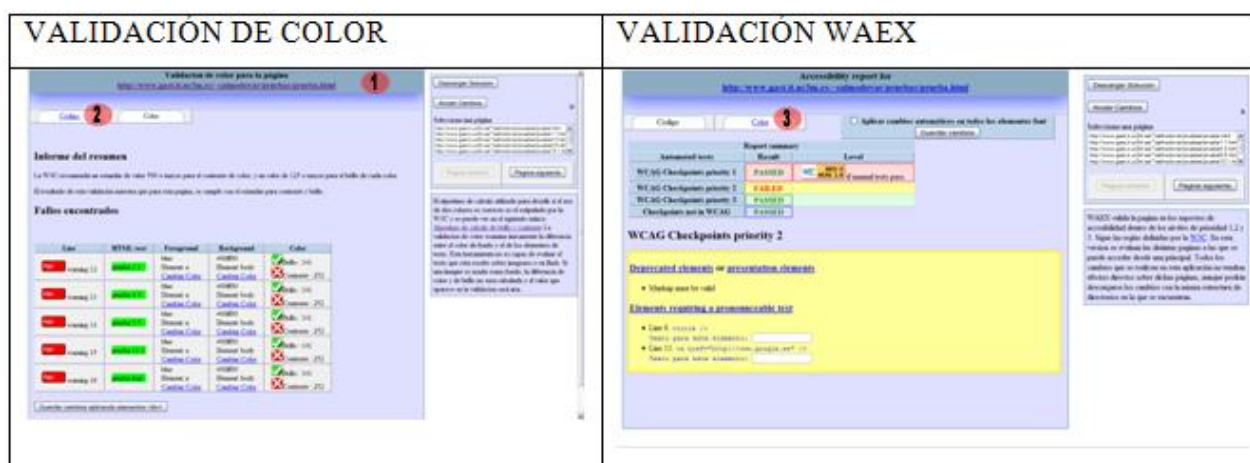


Figura 31 Ejemplo de salidas de WAEX2

Enlace 1: Este primer enlace indica la página que se ha validado y de la que se muestran los resultados en la parte inferior. El enlace da acceso al código de la página con los posibles cambios que el usuario haya podido realizar.

Enlace 2: Manda realizar la validación tipo WAEX [1] desde la actual (la de color).

Enlace 3: Manda realizar la validación de color desde la actual (WAEX [1]).

En ambos casos se puede distinguir la parte de validación que corresponde con la parte izquierda, de las funcionalidades, que aparecen en la parte derecha de la pantalla.

En la parte de las funcionalidades tenemos los botones de “Página Siguiente” y “Página Anterior” que realizará la petición de validación de la siguiente o anterior página de la lista (Figura 32). También puede seleccionarse la página que se desea validar desde la lista que aparece por encima de estos botones:

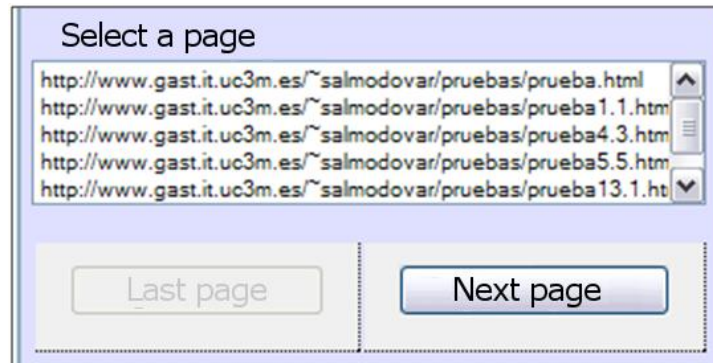


Figura 32 Selección de una página para ser validada

El botón del menú de la derecha cuyo texto es “Descargar” le da la opción al usuario de descargarse las páginas que han sido validadas y/o modificadas a su propio ordenador.

1.1.2. Realizar Cambios de color

Los cambios de color de las páginas únicamente podrán realizarse sobre aquellos elementos en los que el validador haya detectado algún error. Se muestra en el resultado de validación una tabla con una fila por cada error, indicando la línea, contenido, color

de la fuente y color del fondo. En la última columna, tal como se muestra en la siguiente imagen (Figura 34), se muestra el resultado de la validación de este error, comprobando por cada uno de ellos dos propiedades: el brillo y el contraste. El resultado de cada propiedad validada es mostrado mediante un icono, que será un tic de color verde si es correcto el uso de dichos colores para esa propiedad, o uno rojo si no lo es, tal y como se muestra en la imagen de la Figura 33 Leyenda de la validación de colores.. Para que la validación de dos colores sea correcta, ambas propiedades deben mostrar un símbolo verde.

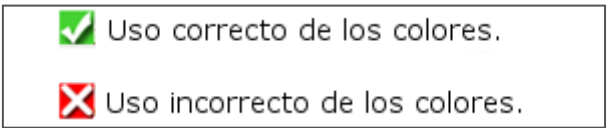


Figura 33 Leyenda de la validación de colores.

Para modificar el color de un elemento debe dirigirse a la fila de la tabla donde se encuentra el elemento que desea modificar el color de la fuente utilizada, haga clic en “Cambiar Color” de la columna Foreground. Si en cambio se desea modificar el color del fondo en el elemento, se deberá hacer clic sobre el enlace de la columna Background. En ambos casos aparece un selector de color para ayudar a elegir el deseado. Una vez seleccionado uno, se actualizará la validación del elemento actualizando si el par de colores seleccionado es válido o no.

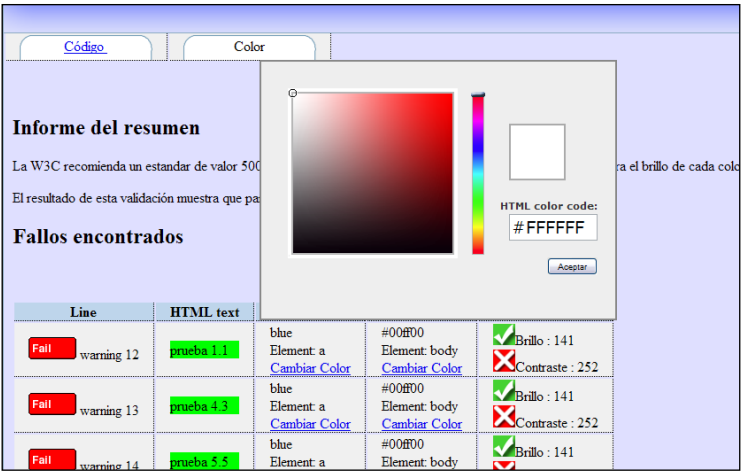
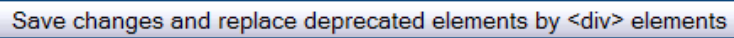



Figura 34 Ejemplo para modificar colores de una página

Una vez haya realizado los cambios oportunos y desee hacerlos efectivos para esta página, pulse sobre el botón:

A rectangular button with a light blue gradient and a thin border. The text inside is "Save changes and replace deprecated elements by <div> elements" in a small, dark font.

1.1.3. Realizar Cambios de Validación del Código

En todas las validaciones que se realizan a través de WAEX [1], en esta aplicación aparecerán en las opciones para las que está habilitado, un campo donde se pueda añadir el texto oportuno para el elemento en el que se ha generado un error. Estos datos se suelen meter como texto en todos ellos, excepto en el que da el error de la WCAG5.5 para el que es necesario seleccionar un color de entre todos los disponibles.

Cada error con la propiedad de corrección online incluye a su lado una entrada de datos. Para hacer que la página los tenga en cuenta y guarde su estado para futuras validaciones o su propia descarga, haga clic sobre el botón 

GLOSARIO DE TÉRMINOS

Acrónimos

CSS

Cascading Style Sheet.

Hoja de estilo en cascada. [9]

DTD

Document Type Definition.

Definición del tipo de documento.

HTML

Hipertext Mark-up Language.

Lenguaje de marcado de hipertexto. [7]

URL

Uniform Resource Locator.

Localizador de recurso único, es decir, la dirección única que identifica a una página web en Internet.

W3C

World Wide Web Consortium (W3C [3])

WAEX

Web Accessibility Evaluator in a single XSLT file.

Evaluados de accesibilidad Web en un único fichero XSLT.

WCAG

Web Content Accessibility Guidelines.

Directrices de Accesibilidad del Contenido Web.

XHTML

Extensible Hypertext Markup Language

Lenguaje de marcado de hipertexto extensible. [8]

XSL

Extensible Stylesheet Language.

Lenguaje de hoja de estilo extensible.

XSLT

Transformations XSL.

Transformaciones XSL.

Definiciones

Comprobaciones automáticas

Son aquellas realizadas mediante una aplicación informática que analiza el código de una página web, devolviendo una serie de anotaciones con los fallos encontrados.

Comprobaciones manuales

Son las reglas que deben tener en cuenta los desarrolladores y que solo ellos pueden detectar. Son imposibles de detectarlas mediante un sistema informático.

Comprobaciones semiautomáticas

Guían al usuario a la hora de facilitarle la evaluación pero no dejan de delegar en su juicio personal esta validación.

Cookies

Fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas. En ocasiones también se le llama "huella".

GET

Método de envío de variables a través de la URL. [17]

Open Source

Término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

POST

Método de envío de variables a través de la entrada estándar STDIO.

Software

Palabra procedente del inglés sin traducción exacta en español, y reconocida por ello por la Real Academia Española de la Lengua. Su significado en este proyecto es referente a un programa o aplicación.

Validador de accesibilidad

Son las herramientas utilizadas para comprobar el cumplimiento de los estándares web de la W3C [3].

REFERENCIAS

Documentos electrónicos

[1] Luque Centeno, Vicente. Web Accessibility Evaluator in a single XSLT file

[<http://www.it.uc3m.es/vlc/waex.html>]. (Consultado 16/12/2009).

[2] Luque Centeno, Vicente. WAEX XSLT file (wcag.xsl)

[<http://www.it.uc3m.es/vlc/wcag.xsl>]. (Consultado 31/10/2009).

[3] W3C. (11/01/2010). World Wide Web Consortuim (W3C).

[<http://www.w3c.es/>]. (Consultado 13/01/2010).

[4] W3C. (5/5/1999). Web Content Accessibility Guidlines 1.0.

[<http://www.w3.org/TR/WCAG10/>]. (Consultado 7/1/2010).

[5] W3C. (11/12/2008). Web Content Accessibility Guidelines 2.0.

[<http://www.w3.org/TR/WCAG/>]. (Consultado 7/1/2010).

[6] W3C. (2006). Advanced Search for Web Accessibility Evaluation Tool

[<http://www.w3.org/WAI/ER/tools/advanced>]. (Consultado 10/11/2008).

[7] HTML Hypertext Markup Language.

[<http://www.w3schools.com/tags/default.asp>]. (Consultado 8/1/2010).

[8] XHTML eXtensible Hypertext Markup Language.

[<http://es.wikipedia.org/wiki/XHTML>]. (Consultado 8/1/2010).

[9] W3Schools. (2009). CSS. Cascade Style Sheet

[http://www.w3schools.com/CSS/CSS_reference.asp]. (Consultado 8/1/2010).

- [10] W3C. (2000). Techniques For Accessibility Evaluation And Repair Tools. Checkpoint 2.2.
[<http://www.w3.org/TR/2000/WD-AERT-20000426>]. (Consultado 7/1/2010).
- [11] W3C. (1999). Web Content Accessibility Guidelines 1.0. Guideline 1.
[<http://www.w3.org/TR/WCAG10/#gl-provide-equivalents>]. (Consultado 8/1/2010).
- [12] W3C. (1999). Web Content Accessibility Guidelines 1.0. Guideline 2.
[<http://www.w3.org/TR/WCAG10/#gl-color>]. (Consultado 8/1/2010).
- [13] W3C. (1999). Web Content Accessibility Guidelines 1.0. Guideline 4.
[<http://www.w3.org/TR/WCAG10/#gl-abbreviated-and-foreign>] (Consultado 8/1/2010).
- [14] W3C. (1999). Web Content Accessibility Guidelines 1.0. Guideline 5.
[<http://www.w3.org/TR/WCAG10/#gl-table-markup>]. (Consultado 8/1/2010).
- [15] W3C. (1999). Web Content Accessibility Guidelines 1.0. Guideline 12.
[<http://www.w3.org/TR/WCAG10/#gl-complex-elements>]. (Consultado 8/1/2010).
- [16] W3C. (1999). Web Content Accessibility Guidelines 1.0. Guideline 13.
[<http://www.w3.org/TR/WCAG10/#gl-facilitate-navigation>]. (Consultado 8/1/2010).
- [17] Haas, Juergen. GET Linux command. *GET comando linux*.
[http://linux.about.com/library/cmd/blcmdl1_GET.htm]. (Consultado 29/12/2009).
- [18] GNU Wget 1.12 Manual del comando WGET
[<http://www.gnu.org/software/wget/manual/wget.html>]. (Consultado 29/12/2009).
- [19] Access Keys (2004-2010). AccessColor. *Validador de color Web*.
[<http://www.accesskeys.org/tools/color-contrast.html>]. (Consultado 12/12/2009).
- [20] W3Schools. JAVASCRIPT
[<http://www.w3schools.com/js/default.asp>]. (Consultado 8/1/2010).
- [21] Mozilla. (2009). XUL.
[<https://developer.mozilla.org/En/XUL>]. (Consultado 8/1/2010).

- [22] Yahoo! Inc. (2010) PHP
[\[www.php.net/downloads.php\]](http://www.php.net/downloads.php). (Consultado 8/1/2010).
- [23] Soulie, Juan. (2009) C++
[\[http://www.cplusplus.com/doc/tutorial/\]](http://www.cplusplus.com/doc/tutorial/). (Consultado 8/1/2010).
- [24] Google. (2009). Google Traductor.
[\[http://translate.google.com/translate_tools?hl=es&layout=1&eotf=1\]](http://translate.google.com/translate_tools?hl=es&layout=1&eotf=1).
(Consultado 13/01/2010).
- [25] (2007). Reconocedor de idiomas.
[\[http://www.chuidiang.com/java/codigo_descargable/idiomas/idioma.php\]](http://www.chuidiang.com/java/codigo_descargable/idiomas/idioma.php). (Consultado 13/01/2010).
- [26] Pearl Crescent. Pearl Crescent Page Saver.
[\[http://pearlcrescent.com/products/pagesaver/\]](http://pearlcrescent.com/products/pagesaver/). (Consultado 19/01/2010).
- [27] Hovinne, Jean-François. (2005). Editor XHTML basado en Web (WYMeditor)
[\[http://www.wymeditor.org/en/\]](http://www.wymeditor.org/en/). (Consultado 7/1/2010).
- [28] Oqbuji, Uche. (2007). Browser extensions using XUL, Part 1: Create a Firefox browser extension with user-interface features.
[\[http://www.ibm.com/developerworks/web/library/wa-xul1/\]](http://www.ibm.com/developerworks/web/library/wa-xul1/). (Consultado 27/10/2009)
- [29] Fundación CTIC. Test de Accesibilidad Web (t.a.w.)
[\[http://www.tawdis.net/\]](http://www.tawdis.net/). (Consultado 31/10/2009)

Referencias bibliográficas

- [30] Kenneth C. Louden. *Construcción de Compiladores. Principios y práctica*. Thomson, 2004.
- [31] Douglas K. van Duyne. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Addison – Wesley. 2002.
- [32] Castro, Elisabeth. *HTML con XHTML y CSS*. Anaya Multimedia. 2003.
- [33] Schmitt, Christopher. *CSS cookbook*. O'Reilly. 2004.

Herramientas

- HTML [7]
- CSS [9]
- JAVASCRIPT [20]
- XUL [21]
- PHP [22]
- C++ [23]